AN ABSTRACT OF THE DISSERTATION OF

Todd Kulesza for the degree of Doctor of Philosophy in Computer Science presented on December 1, 2014.

Title: Personalizing Machine Learning Systems with Explanatory Debugging

Abstract approved: _____

Margaret M. Burnett

How can end users efficiently influence the predictions that machine learning systems make on their behalf? Traditional systems rely on users to provide examples of how they want the learning system to behave, but this is not always practical for the user, nor efficient for the learning system. This dissertation explores a different personalization approach: a two-way cycle of explanations, in which the learning system explains the reasons for its predictions to the end user, who can then explain any necessary corrections back to the system. In formative work, we study the feasibility of explaining a machine learning system's reasoning to end users and whether this might help users explain corrections back to the learning system. We then conduct a detailed study of how learning systems should explain their reasoning to end users. We use the results of this formative work to inform Explanatory Debugging, our explanation-centric approach for personalizing machine learning systems, and present an example of how this novel approach can be instantiated in a text classification system. Finally, we evaluate the effectiveness of Explanatory Debugging versus a traditional learning system, finding that explanations of the learning system's reasoning improved study participants' understanding by over 50% (compared with participants who used the traditional system) and participants' corrections to this reasoning were up to twice as efficient as providing examples to the learning system.

Personalizing Machine Learning Systems with Explanatory Debugging

by

Todd Kulesza

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 1, 2014
Commencement June 2015

Doctor of Philosophy dissertation of Todd Kulesza presented on December 1, 2014.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Todd Kulesza, Author

ACKNOWLEDGEMENTS

I was fortunate to earn a master's degree under the mentorship of a professor who believed I could go further and encouraged me to pursue a Ph.D., and this work is the result of that confidence. Thank you, Margaret Burnett—your mentorship, encouragement, and patience have helped me grow into a more precise writer, a more critical thinker, and a more capable researcher.

During my time in graduate school I have collaborated with many wonderful researchers. I would especially like to thank Simone Stumpf, Saleema Amershi, Scott Fleming, Irwin Kwan, Chris Bogart, and Eric Walkingshaw for the pleasure of working with and learning from each of you, as well as my Ph.D. committee for your feedback and insights: Weng-Keen Wong, Carlos Jensen, Alex Groce, and Maggie Niess.

Finally, none of this work would have been possible without the support of family and friends. I'd particularly like to acknowledge Bill and Iris McCanless, at whose cafe many of these chapters were written; Koa Tom, for three years worth of understanding and encouragement, plus an uncanny ability to get me out of the lab and into the wider world when I've most needed it; and my parents and grandparents, for instilling a love of learning from an early age, and the sacrifices they made to improve their children's educations. Thank you.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1: Introduction

1.1 Motivation

Machine learning systems have become a part of everyday life for hundreds of millions of people. Netflix, for example, uses machine learning techniques to recommend movies to its 50 million[1] subscribers. Pandora uses machine learning to create custom playlists for its 76 million[2] listeners. Google employs machine learning to organize and filter junk mail from the inboxes of its 425 million[3] Gmail users, and Facebook uses machine learning to personalize content for its 1.3 *billion*[4] users.

Advances in machine learning have made these new types of applications possible, but the widespread use of programs that respond differently to each user also brings a novel challenge: how can end users effectively control the predictions or recommendations that these learning systems make on their behalf?

For example, if end user Alice notices that many of the email messages she receives each day are junk mail, she typically has but one option: mark each one as *junk mail* in her email software. The software likely gives her no feedback about what happened after she marks each message as *junk*, but behind the scenes it has adjusted its reasoning for detecting messages that Alice will think are junk mail. Ideally this reasoning will perfectly match Alice's concept of junk mail, but in reality it almost certainly will not—the computer does not know *why* Alice decided to label some messages as junk mail and other messages as legitimate mail, so it must infer her reasoning based on her labeled messages. Indeed, this reliance on machine inference of human reasoning means that Alice will need to keep telling the software about each of its mistakes and hoping that it performs better in the future. No professional software developer would be satisfied with such a slow and imprecise approach to fixing software that is not

[1] Shareholder report, Q2'14: `http://bit.ly/1vWUIXM`
[2] Financial results, Q2'14: `http://bit.ly/1DtkhAF`
[3] Google I/O 2012: `http://bit.ly/1zi95Iu`
[4] Shareholder report, Q2'14: `http://bit.ly/1FrVuz9`

operating as desired, so why do we assume end users will be able to use it to effectively and efficiently control the myriad machine learning systems they interact with each day?

Instead of this guess-and-check approach to controlling machine learning systems, what if end users could directly view and adjust the learning system's reasoning, similar to how professional software developers would debug a program's source code? If successful, users would gain new power over their machine learning systems—users could personalize each system to match their idiosyncratic tastes, and perhaps even apply them to situations the original developers had never intended.

This dissertation presents just such an approach: Explanatory Debugging. In Explanatory Debugging, the learning system explains the reasons for its predictions to its end user, who in turn explains corrections back to the system. We hypothesize that this cycle of explanations will help users build useful *mental models*—internal representations that allow people to predict how a system will behave (Johnson-Laird, 1983)—that in turn will allow users to explain any necessary corrections back to the learning system better and faster than they could with a traditional black box instance-labeling system.

1.2 Thesis statement

> *Two-way explanations will allow people to better personalize machine learning systems by helping them build better mental models of the learning system and allowing them to directly correct mistakes in the learning system's reasoning.*

Just as professional software developers need a useful mental model of the software they work on, end users attempting to personalize a machine learning system will need to develop a useful mental model of the learning system. This mental model will help the user understand why the system is behaving in a given manner and allow the user to predict how the system will respond if certain changes are made to it.

Mental models by themselves, however, are insufficient—users also need to be able to *act* upon their knowledge. Thus, learning systems need to both explain their reasoning and allow the user to directly adjust this reasoning. We hypothesize that by being able to accurately predict how the system will respond to specific adjustments, the user will be able to explain corrections to the system more successfully than a user whose mental model is flawed.

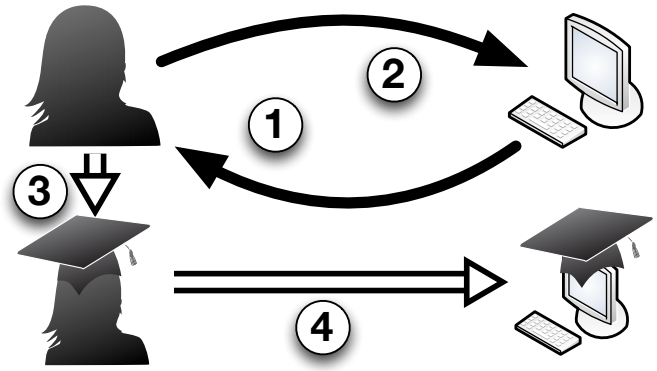


Figure 1.1: The Explanatory Debugging approach for users to (1) learn about the system's current reasoning and (2) interactively adjust it as needed. In the process, the user is building a better mental model of how the learning system operates, allowing the now-smarter user to more effectively personalize the system (3), with the eventual outcome of a "smarter" machine learning system (4).

Further, because interactive machine learning involves a continuous cycle of input from the user and output from the learning system, there is a ready-made opportunity to teach users about the learning system's reasoning in the context of each output. If the user disagrees with any of these outputs, she can correct the system and immediately see the results of her effort, further refining her mental model and, hopefully, improving the effectiveness of any future explanations she provides to the learning system. This cycle is illustrated in Figure 1.1.

1.3 Terminology

The work presented in this dissertation lies at the intersection of two fields of computer science: human-computer interaction (HCI) and machine learning (ML). As readers from one discipline may be unfamiliar with terminology from the other, the following list defines several common terms used throughout this dissertation.

Machine learning system A computer program (or part of a computer program) that uses one or more algorithms to "learn" reasoning based on user-supplied inputs. Different types of machine learning systems are referred to by a variety of different

names; this dissertation will focus on *classifiers* (learning systems that attempt to group items into categories) and *recommenders* (learning systems that attempt to rank items by certain criteria, such as desirability).

Interactive machine learning A machine learning system that rapidly updates its outputs (e.g., its predictions or recommendations) in response to new user-provided inputs (e.g., ratings or labels). As discussed in Amershi et al. (in press), interactivity is best viewed as a continuum; highly interactive machine learning systems may update their predictions in real-time after each user interaction, while less interactive systems update their predictions less frequently.

Personalization The act of providing feedback to an interactive machine learning system with the intent of changing its predictions to more closely align with a particular end user's reasoning.

Instance labeling A traditional method for personalizing interactive machine learning systems. An *instance* refers to the input that the learning system makes predictions about; a *label* is the output classification the user wants associated with the input. Examples include marking email messages (instances) as junk mail (label); telling an Internet radio station that you liked (label) the song (instance) it just played; and telling photo-tagging software that the face (instance) it tagged as you is actually someone else (label).

Feature An aspect or measurement of an instance that a machine learning system uses when making predictions or recommendations. For example, text classifiers often use the presence or absence of specific words as features, while a music recommender's features might include the number of times a group of users have played each song.

Mental model A mental representation of a real-world system that governs how the person holding the mental model will interact with the system (Johnson-Laird, 1983). Famous examples include peoples' decisions of whether to first try pushing or pulling to open a specific door (Norman, 2002) and dialing a thermostat higher in the hope of heating a home *faster* rather than making it *hotter* (Kempton, 1986).

1.4 Potential use cases

We believe that giving users the option to learn why a machine learning system made a given prediction—and correct it if necessary—would be beneficial under any circumstances. Whether users then choose to attend to the system's explanations and provide corrective feedback to the learning system would depend on each user's perception of the benefits (e.g., a more useful machine learning system) versus the costs (e.g., time spent attending to explanations) and risks (e.g., the possibility they will not make their learning system more useful, or worse, make it *less* useful). If, for example, explanations were only shown on-demand, the user experience for people satisfied with the learning system's predictions would not be harmed. Unsatisfied users, however, would have the possibility of viewing the explanations to help them personalize the system. Further, there are specific cases where empowering users with two-way explanations may be especially helpful:

1. **When personalization must happen quickly.**
   There are circumstances where users need to personalize a machine learning system quickly, such as someone preparing a personalized Internet radio station for a party she is hosting that evening. Training the system by rating each song it plays would be impractical during the party and time-consuming before it. However, researchers have found that directly specifying the features a text classifier should pay attention to is five times more efficient (in terms of time spent) than labeling instances (Raghavan et al., 2006); if this efficiency holds in other domains, it could allow the host to quickly personalize an Internet radio station by feeding it precise instructions about the types of songs to play, rather than feeding it numerous examples and hoping the learning system correctly identifies song aspects the user feels will be appropriate for her event.

2. **When sufficient training data does not exist.**
   Sometimes the user of a machine learning system wants to begin using the system before it has seen a sufficient amount of instances to learn from, but the accuracy of such systems is highly erratic (Brain and Webb, 1999). For example, a parent who has recently joined the PTA at his child's school may want all PTA-related emails automatically filed to a specific folder, but he doesn't yet have examples of such

emails. However, if the parent knew the email classifier worked by examining each message's words, he could prime the classifier with words like "PTA", the name of his child's teacher, and the name of the school. A similar problem often exists in anomaly detection systems, where examples of anomalies are too infrequent to provide sufficient training data (Chandola et al., 2009).

3. **When the user's concept has evolved.**
   Even machine learning system with sufficient training data can run into problems when the user's concept of how items should be classified changes (Kulesza et al., 2014). Consider a user who has spent months personalizing a news aggregator by continually giving it positive feedback when it recommended news articles about local politics. What happens when she moves to a new city? She could spend months providing additional examples of local political articles to counter-balance the existing information the classifier has learned from, but even so, the classifier may "learn" that she's interested in political news in *both* regions. Telling the classifier—via instance labeling—that political stories from her prior city are no longer desired is also risky, as it may make the learning system stop recommending political stories entirely. A much simpler approach would be to view the classifier's reasoning, identify any parts of this reasoning related to her prior city, and update them to match her new home.

4. **When a better understanding of the learning system matters.**
   When machine learning systems assist with important tasks, users need to understand what the learning system can do reliably versus when its predictions are likely to need manual verification (Groce et al., 2014). For example, consider an aging-in-place system that uses machine learning techniques to determine if everything looks normal, or whether a caretaker should be notified to check in. If the system uses motion sensors to identify activity but the elderly person has a pet, the motion sensor data may not be as meaningful as in other homes. By understanding such aspects of the system, the user (here, a caretaker or family member) can then explain to the system that it should ignore or place less importance on motion sensor data.

1.5 Proposed contributions

This dissertation makes the following contributions:

- An understanding of the potential impact that a good mental model may have on an end user's ability to personalize a machine learning system that supports two-way explanations.
- An understanding of how attributes of explanations and their intelligibility types (as defined in Lim and Dey, 2009) impact the development of end users' mental models.
- Explanatory Debugging: A novel approach, grounded in theory and our empirical data, to help end users build useful mental models of a machine learning system and allow them to personalize its outputs via two-way explanations.
- An empirical evaluation of Explanatory Debugging in the domain of text classification.

Chapter 2: Background and literature review

2.1 Mental models

The feasibility of Explanatory Debugging rests on explanations—explanations to help the user understand the machine learning system, and explanations from the user to correct the learning system's mistakes. Thus, if users were unable to understand the learning system, our entire approach would fail. Machine learning systems are complex, but psychologists have developed a theory describing how people reason about complex systems: the mental model theory of thinking and reasoning (Johnson-Laird, 1983).

Mental models are internal representations that people generate based on their experiences in the real world. These models allow people to understand, explain, and predict phenomena, with an end result of being able to then act accordingly (Johnson-Laird, 1983). Understanding how end users build mental models of learning systems—and how these models evolve over time—may allow researchers to design more effective explanations of a system's reasoning than an ad hoc approach.

The contents of mental models can be concepts, relationships between concepts or events (e.g., causal, spatial, or temporal relationships), and associated procedures. For example, an end user's mental model of how a computer works may be as simple as a screen that displays everything typed on a keyboard, and the sense that it "remembers" these things somewhere inside the computer's casing. Mental models can vary in their richness—an IT professional, for instance, likely has a much richer mental model representing how a computer works than the above example. According to Norman, these models do not have to be entirely complete (i.e., encompass all relevant details) to be useful, but they must be sound (i.e., accurate) enough to support effective interactions (Norman, 1987).

The varying richness of mental models results in a useful distinction: functional (shallow) models imply that the end user knows how to use the system, but not how it works in detail, whereas structural (deep) models provide a detailed understanding of how the system works (Rogers et al., 2011). In this dissertation we are primarily

concerned with structural models because these enable users to predict how changes to the system will alter its outputs. Theoretically, a structural mental model that accurately reflects a system will help the user to predict how his or her actions will alter the system's outputs, thus allowing the user to precisely control the system. Conversely, users with flawed structural models are likely to experience difficulty predicting how their actions influence the system's outputs, which in turn will lead to difficulty controlling the system's behavior. Many instances of inaccurate mental models (both functional and structural) guiding erroneous behavior have been observed in tasks as varied as opening doors to controlling thermostats (Jonassen and Henning, 1996; Norman, 1987; Kempton, 1986).

Mental models develop "naturally" due to repeated use of or exposure to a system over time, but models that develop through use may be highly inaccurate. For example, Kempton estimates that as many as 50% of Americans erroneously believe their home furnace operates similar to a valve or faucet—that turning the thermostat higher causes more hot air to come out of the furnace, thus heating up a room faster than a lower thermostat setting (Kempton, 1986). The source of this misunderstanding, he contends, is the abundance of devices that *do* operate using valves in our daily environments (e.g., water faucets, gas burners, automobile pedals), and the scarcity of familiar devices that behave similarly to thermostats. Providing explanations of how such unfamiliar or complex systems operate can avoid this problem, but introduces a new challenge: how can a complex system convince end users to pay attention to its explanations?

Theory suggests that the challenge of engaging user attention can be overcome by communicating the benefits of learning more about the complex system. Blackwell's Attention Investment model hypothesizes that people decide the level of effort to expend on a given task based on their perceptions of cost, risk, and eventual benefit (Blackwell, 2002). According to this model, users will be more likely to invest time toward understanding a system when they believe the benefits (e.g., an improved system) will outweigh the expected costs (e.g., time and effort spent) and possible risks (e.g., being unable to improve the system, or potentially breaking it).

To verify that our approach does attract user attention and succeeds in helping users build better structural mental models, we will need a method for quantifying mental models. However, capturing in-the-head representations—such as mental models—without simultaneously influencing them is a research problem without a simple solution.

The very act of measuring users' mental models is likely to influence those models (Doyle et al., 2008), thus reducing their validity. For example, asking study participants to draw a flowchart of a system may encourage them to think of how information "flows" through a system more than they otherwise would. Norman discusses how verbal or written elicitations may be incongruous with participants' observed actions and will be necessarily incomplete (Norman, 1987). One method for remedying this incompleteness is to provide participants with the various components of a system and ask them to properly arrange or interconnect them, but this transforms the problem from having a bias toward recall to one with a bias toward recognition (Otter and Johnson, 2000). Carley and Palmquist developed a partially automated technique for defining mental models based on textual analysis, but this approach is time consuming and still suffers from problems of incompleteness (Carley and Palmquist, 1992). Thus, capturing incomplete or influenced mental models is always a threat to validity when studying mental models. In Section 3.2.4 we explain how we attempt to work around this issue.

2.2 Explaining machine learning

One method for helping users learn about machine learning systems is to increase the system's transparency, as in the "white box" model proposed by Herlocker et al. (2000). To help people build mental models of a system, it has been argued that the system should be transparent and intuitive, and users should be provided with appropriate instructions of how it works (Rogers et al., 2011). Our own preliminary research has found that users will refine their mental models of a learning system when the system makes its reasoning transparent (Kulesza et al., 2010), but some explanations may lead to only shallow mental models (Stumpf et al., 2007). Alternatively, a system's reasoning can be made transparent via human-to-human instruction, and this can help with the construction of mental models of how it operates (McNee et al., 2003). A different approach, scaffolded instruction, has been shown to contribute positively to learning to use a new static system (Rosson et al., 1990); however, scaffolding instruction for a dynamic system poses challenges because the changing nature of the system means the scaffolding cannot be pre-planned by a human—it must be generated in real-time by the system itself.

Scaffolding is but one example of applying the Minimalist model of instruction, which is designed to help *active users* (users who are interested in accomplishing a goal with a system and only want to learn enough about it to accomplish this goal (Carroll and Rosson, 1987)) learn about a system while performing real work (van der Meij and Carroll, 1998). Because we doubt many users have the primary goal of learning how their classifiers and recommenders operate—as opposed to how to make them behave in a more useful manner—our audience also fits into this active user category. Thus, the principles of Minimalist instruction (e.g., interleave instruction with real work, leverage existing knowledge, tightly couple explanations to the system's current state) may also be appropriate for teaching users about their machine learning systems.

However a learning system explains its reasoning to end users, these explanations will somehow impact users' mental model of the system. Very little work, however, has explored *how* explanations impact the development of user's mental models of machine learning systems. One such study found that prolonged use of a learning system induced plausible (though imperfect) models of its reasoning, and that these models were surprisingly hard to shift, even when people were aware of contradictory evidence (Tullio et al., 2007). In another study, Lim et al. found that explanations of *why* a learning system made a given prediction (or, why it did not make a different prediction) helped users better understand the learning system (Lim et al., 2009), while explanations of what might happen if the user performed a given action did not help improve participants' understanding. Many researchers have studied the impacts of learning system transparency on factors such as satisfaction (Cramer et al., 2008; Bostandjiev et al., 2012; Tintarev and Masthoff, 2012; Sinha and Swearingen, 2002; Herlocker et al., 2000; Billsus et al., 2005), system performance (Thomaz and Breazeal, 2006), and trust (Dzindolet et al., 2003; Glass et al., 2008), but these works did not explore a link between transparency and the quality of users mental models.

Researchers have explored numerous methods to explain common machine learning algorithms, but the impact of these methods on end users' mental models has rarely been evaluated. Much of the work in explaining probabilistic machine learning algorithms has focused on the naive Bayes classifier, often employing visualizations such as pie charts (where each pie slice describes the weight of evidence for a particular feature) (Becker et al., 2001) or bar charts (where each bar's position describes a feature's information gain) (Kulesza et al., 2011). Nomograms, a visualization technique that uses linear scales

to represent variables in an equation, have been proposed for explaining both naive Bayes (Možina et al., 2004) and SVM (Jakulin et al., 2005) classifiers, but again, we do not know what impact these explanations may have on end users' mental models. Poulin et al. employed a stacked bar chart visualization to support the more general class of linear additive classifiers, and used a prototype to successfully help biologists identify errant training samples in their data (Poulin et al., 2006). Lacave and Díez present several approaches, both textual and graphical, for describing general Bayesian networks, but these techniques are too computationally expensive to result in a real-time cycle of explanations with the end user (Lacave and Díez, 2002).

While researchers currently have a limited understanding of the impact that explanations of learning system have on users' mental models, researchers have developed taxonomies we can use in our exploration of explanations. For example, Lacave and Díez enumerated the types of information that learning systems can provide to end users: explanations of evidence (e.g., "The size, shape, and color of this fruit suggest it is an apple"), explanations of the machine learning model (i.e., the learning algorithm itself, devoid of training data), and explanations of the system's reasoning (e.g., precisely how an object's size, shape, and color contributed to its classification as apple). Lim and Dey provide a more nuanced view of a learning system's potential explanations via *intelligibility types* (Lim and Dey, 2009). For example, three intelligibility types cover Lacave and Díez's "explanation of the learning model" category: explanation of the model itself, explanation of the inputs it may use, and explanation of the outputs it could produce (Lim and Dey, 2009). We will return to these intelligibility types in our study of explanations (Chapter 4) and the design of our explanation-centric personalization approach (Chapter 5).

A machine learning system's mistakes—by virtue of being unexpected—may naturally attract user attention and inspire curiosity, causing users to attend to its explanations. For example, Hastie's investigations of causal reasoning found that "when unexpected behaviors are attributed to a person, the perceiver is relatively likely to engage in causal reasoning to understand why these behaviors occurred" (Hastie, 1984). This finding also appears to hold when the unexpected behavior is attributed to a machine: Wilson et al. designed an approach—surprise-explain-reward—that successfully leveraged participants' curiosity about unexplained software behaviors to engage their attention (Wilson et al., 2003). In this study, participants had the choice

to view explanations describing an unfamiliar software testing tool (assertions) and the potential benefits of employing it; nearly all of their participants went on to make use of this tool in response to surprising software behavior (Wilson et al., 2003). Because users are likely to wonder why a machine learning system made a given mistake, such mistakes provide an opportunity to engage the user's attention by explaining why the learning system erred.

2.3 Personalizing machine learning systems

The traditional method for personalizing a machine learning system is to label instances, thus providing it with new training data to "learn" from. In this approach, users label a large number of instances all at once, then "retrain" the learning system using this new data (Amershi et al., in press). By processing data in large batches, however, users are unable to receive feedback indicating how their labels have influenced the learning system's reasoning. For example, if half of their labels improved the system's performance while the other half harmed it, the user may look at the resulting system and conclude that labeling did nothing.

A more interactive personalization approach also exists. Fails and Olsen Jr. first popularized the phrase *interactive machine learning* in a paper describing how an iterative train-feedback-correct cycle allowed users to quickly correct the mistakes made by an image segmentation system (Fails and Olsen Jr., 2003). Since then, researchers have explored using this cycle of quick interactions to train instance-based classifiers (Fogarty et al., 2008; Bryan et al., 2014), enable better model selection by end users (Amershi et al., 2010; Talbot et al., 2009; Fiebrink et al., 2011), elicit labels for the most important instances (e.g., active learning) (Settles, 2010; Cakmak et al., 2010), and to improve reinforcement learning for automated agents (Knox and Stone, 2012). A similar approach has been used by Programming by Demonstration (PBD) systems to learn programs interactively from sequences of user actions (see (Lieberman, 2001) for a collection of such systems). For example, if a user sees the Gamut PBD system make a mistake, they can "nudge" the system in the right direction, which has the impact of adding or removing a training example and showing the user the revised system's output (McDaniel and Myers, 1997). These use cases, however, largely treat the machine learning system as a "black box"—users can try to personalize the system by providing it with different

inputs (e.g., labeled instances), but are unable to see *why* different inputs may cause the system's outputs to change.

Some researchers have sought to expand the types of inputs interactive machine learning systems can process, and in so doing have also increased system transparency. For example, user feedback may take the form of model constraints (Kapoor et al., 2010), critiques to the model's search space (Vig et al., 2011), or adjustments to the model's weights of evidence (Kulesza et al., 2011). In addition to supporting novel user feedback mechanisms, these approaches increase transparency by enabling users to view such facets as the existing model constraints (Kapoor et al., 2010), the model's feature set (Vig et al., 2011; Verbert et al., 2013), or the weights of evidence responsible for each prediction (Kulesza et al., 2011; Bostandjiev et al., 2013). In a similar vein, Amershi et al. bridged active learning (Settles, 2010) with interactive concept learning, creating a mixed-initiative approach that guides users toward identifying helpful training data for learning concepts not directly represented by a classifier's features (Amershi et al., 2009).

Even when novel types of feedback mechanisms do not increase transparency, there are additional reasons to support multiple types of feedback. For example, some problem domains lack sufficient training data for traditional labeling approaches to be effective; in these circumstances, allowing end users to directly specify the features a learning system should use for classification can be more efficient than instance labeling (Das et al., 2013; Raghavan et al., 2006). Instance labeling approaches are also at the mercy of external forces—they require appropriate training examples to be available when the user needs them and suffer from class imbalance problems in "bursty" (e.g., email classification). Further, in domains such as recommendation, there is no single correct answer for the learning system to predict. In such cases, it can be beneficial to allow users to explore the search space via critique-based feedback (Glowacka et al., 2013; Vig et al., 2011; Parra et al., 2014)

A user-centric perspective on machine learning would not require that a sufficient number of appropriate instances exist in order for the user to make their learning systems behave as desired—the instance-based approach takes power away from end users and makes them reliant on the population of extant instances. Instead, Stumpf et al. have argued that if an end user wants to tell the computer how to behave, he or she should have the choice of *explaining* the desired behavior rather than being forced to find

an example of it (Stumpf et al., 2009). EnsembleMatrix (Talbot et al., 2009) is one system that supports such direct manipulations, providing users with both a visualization of a classifier's accuracy and the means to adjust its reasoning; however, EnsembleMatrix is targeted at machine-learning experts developing complex ensemble classifiers, rather than end users working with the deployed classifiers.

Regardless of the feedback mechanism, successfully personalizing a machine learning system can be challenging for end users. Some explorations of novel feedback mechanisms have found that users' attempts to adjust their learning systems' reasoning caused their performance to decrease or fluctuate wildly (Stumpf et al., 2008; Kulesza et al., 2010). Part of the problem is the complexity of learning systems; our own prior work has explored the barriers end users encounter when trying to personalize a learning system, finding that the two most common problems involve identifying which of a learning system's many features should be altered in order to change a specific prediction, and coordinating how such a change will impact other predictions (Kulesza et al., 2011). Even when user feedback is restricted to the traditional instance-labeling approach, users often struggle to label similar items in a similar manner (Kulesza et al., 2014), thus providing conflicting evidence for the learning system to "learn" from. Research into how end users attempt to understand and control complex computer systems, however, already exists in the field of end-user programming, and we next look to it for inspiration in helping end users better personalize machine learning systems.

2.4 Machine learning and end user programming

Because personalizing a machine learning system involves adjusting the system's reasoning, it is very similar to traditional software debugging: if the learned behavior is incorrect, the user needs to identify why and provide a fix. Thus, we view personalizing a machine learning system as an *end-user debugging* problem, and this view suggests that approaches to support personalization should build upon existing end-user programming research.

Machine learning systems are formally tested prior to deployment by machine learning specialists using statistical methods (Hastie et al., 2009), but such methods do not guarantee the system will continue to work satisfactorily as it continues to "learn" from its user's behavior. Indeed, as argued in Groce et al. (2014), after a learning system

has been deployed, *only* the end user is in a position to test it—once its reasoning has been updated (e.g., by the user labeling additional training instances), the original developers no longer have a testable variant that matches the end user's. Systematic testing for end users was pioneered by the What You See Is What You Test approach (WYSIWYT) for spreadsheet users (Rothermel et al., 2001). To alleviate the need for users to conjure values for testing spreadsheet data, "Help Me Test" capabilities were added; these either dynamically generate suitable test values (Fisher II et al., 2006) or back-propagate constraints on cell values (Abraham and Erwig, 2006). Statistical outlier finding has been used in end-user programming settings for assessment, such as detecting errors in text editing macros (Miller and Myers, 2001), inferring formats from a set of unlabeled examples (Scaffidi, 2007), and to monitor on-line data feeds in web-based applications for erroneous inputs (Raz et al., 2002). These approaches use statistical analysis and interactive techniques to direct end-user programmers' attention to potentially problematic values, helping them find places in their programs to fix. Our preliminary work has explored the utility of such approaches when testing machine learning systems, finding they helped participants discover more of a system's failures and test more of the system's logic than regular ad hoc assessment alone (Groce et al., 2014).

A number of debugging approaches leverage systematic testing to help end users find and understand the causes of faulty behavior. For example, in the spreadsheet domain, WYSIWYT allows users to test spreadsheet formulas by placing checkmarks beside correct outputs and X-marks beside incorrect outputs (Rothermel et al., 2001). A fault localization device then traces the data "flowing" into these cells, helping users locate cells whose formulas are likely to be faulty. Woodstein serves a similar purpose in the e-commerce domain: this approach helps users to debug problems by explaining events and transactions between e-commerce services (Wagner and Lieberman, 2004). These approaches exemplify successful attempts to help end users first identify, and then understand the cause of, program failures. To facilitate such understanding, they work to draw a user's attention to the faulty regions of a program's logic. The Whyline (Ko and Myers, 2008) performs a similar function for more traditional programming languages, and also dynamically generates explanations of a program's behavior. Because of the similarity between what the Whyline does for traditional programming languages and

what we hope Explanatory Debugging will do for machine learning systems, we next discuss the Whyline in some detail.

The Whyline pioneered a method to debug certain types of programs in an explanation-centric way (Ko and Myers, 2008). The Whyline was explored in three contexts, encompassing both end user programmers and professional developers: (1) event-based virtual worlds written in the Alice programming system (Ko, 2006), (2) Java programs (Ko and Myers, 2008), and (3) the Crystal system for debugging unexpected behaviors in complex interfaces (Myers et al., 2006). In each case, these tools help programmers understand the causes of program output by allowing them to select an element of the program and receive a list of why and why not questions and answers in response. These "Why?" questions and answers are extracted automatically from a program execution history, and "Why not" answers derive from a reachability analysis to identify decision points in the program that could have led to the desired output. In the Crystal prototype, rather than presenting answers as sequences of statement executions, answers are presented in terms of the user-modifiable input that influenced code execution. In all of these Whyline tools, the key design idea is that users select some output they want to understand, and the system explains the underlying program logic that caused it.

Part of our preliminary research involved translating the Whyline's design concept to the domain of machine learning systems (Kulesza et al., 2011). This built upon the work of Stumpf et al., who explored end-user debugging techniques for text classification systems. Their research began by investigating different types of explanations, as well as user reactions to these explanations (Stumpf et al., 2007); user studies later confirmed that even simple corrections from end users have the potential to increase the accuracy of a learning system's predictions (Stumpf et al., 2008, 2009). For some participants, however, the quality of the system's predictions actually decreased as a result of their corrections—there were barriers preventing these users from successfully personalizing the learning system's reasoning. We later conducted a study building upon this work, identifying and categorizing the barriers end users encountered and the information they requested to overcome them (Kulesza et al., 2011). This exploration taught us that a direct translation of the Whyline to the machine learning domain was impractical, but it did suggest that a different explanation-centric approach may work—if, that is, we can help users overcome the high number of barriers they encounter while attempting to personalize a machine learning system. The rest of this dissertation explores the

development of just such an approach by studying how explanations and feature-based feedback can help users overcome barriers and successfully personalize machine learning systems.

Chapter 3: Exploring the effects of mental model fidelity

3.1 Introduction

At one time or another, most everyone has tried to open a door by pushing it outward when they actually needed to pull it inward, or vice versa. This is the canonical example of a mental model failure in Norman's *The Design of Everyday Things*, in which Norman argues that the design of objects provide hints to users about how to interact with them. For example, seeing a vertical handle on a door suggests it should be pulled, while a plate or horizontal bar suggests pushing (Norman, 2002). These designs gives users clues as they (often unconsciously) build a mental model of what will happen if they pull or push on the door, and if the design is successful, then users' mental models will be correct—they will push doors that open outward and pull doors that open inward. The key point is that even in everyday situations, mental models matter—they inform all of our actions by allowing us to predict the *results* of those actions.

Our explanation-centric approach for personalizing machine learning is predicated upon this same idea: that a user with an accurate mental model of the learning system will be able to better predict how the system will respond to his or her actions than a user with a poor mental model of the system. As discussed in Section 2.2, however, researchers have not yet studied how a user's mental model impacts their ability to personalize a machine learning system—perhaps the amount of time or effort it takes to build a useful mental model of a learning system is impractically high, or the benefit is relatively modest. Thus, we begin by studying the feasibility of helping end users build accurate structural mental models (i.e., how it works) of a learning system and the impact that these models have on end users, as compared with a basic functional mental model (i.e., how to use it).

When discussing how accurately a structural mental model reflects a machine learning system, we use the term *fidelity*. A high-fidelity mental model is a more accurate reflection of the real-world system than a low-fidelity mental model.[1]

The complexity of machine learning systems presents an important question: can users without a computer science background quickly grasp the complexities of a learning system? In the (limited) prior work in this area, users created *plausible* models of how the learning system may work, but these mental models did not necessarily reflect the actual learning system used in the study (Tullio et al., 2007). If it is feasible to help end users build high-fidelity mental models of a learning system, we are interested not only in whether these mental models help users personalize a learning system better than participants who lack such models, but also the potential costs that acquiring these mental models may entail. In particular, we are concerned that the complexity of the system may prove discouraging or anxiety-inducing to some end users, thus resulting in an poor overall user experience. Taken together, these concerns yield the following four research questions:

RQ3.1: *Feasibility*: Can end users quickly build and recall a high-fidelity structural mental model of a learning system's operation?

RQ3.2: *Personalization*: Do end users' mental models have a positive effect on their ability to personalize a machine learning system?

RQ3.3: *Confidence*: Does building a high-fidelity mental model of a machine learning system improve end users' computer self-efficacy and reduce computer anxiety?

RQ3.4: *User Experience*: Do end users with high-fidelity mental models of a machine learning system experience interactions with it differently than users with low-fidelity models?

3.2 Empirical study

To explore the effects of mental model fidelity on end-user personalization of machine learning systems, we needed a domain that participants would be motivated to both

[1] We say *fidelity* instead of *accuracy* for consistency with our analysis of explanations in Chapter 4, in which we show that fidelity is impacted by (1) the presence of information (referred to as *completeness* in Chapter 4) and (2) whether said information is accurate (referred to as *soundness* in Chapter 4).

use and personalize. Music recommendations, in the form of an adaptable Internet radio station, meet these requirements, so we created an Internet radio platform (named AuPair) that users could personalize to play music fitting their particular tastes.

Mental models may continue to evolve while users interact with a system, and this evolution would be difficult to capture in a brief laboratory experiment. Thus, to reflect the fact that many machine learning systems are used over periods of days, weeks, or months, rather than minutes or hours, we designed our experiment to combine a controlled tutorial session in a laboratory with an uncontrolled period of field use. The study lasted five days, consisting of a tutorial session and pre-study questionnaires on Day 1, then three days during which participants could use the AuPair prototype as they wished, and an exit session on Day 5.

3.2.1 AuPair Radio

AuPair allows the user to create custom *stations* and personalize them to play a desired type of music. Users start a new station by seeding it with a single artist name (e.g., "Play music by artists similar to Patti Smith", Figure 3.1). Users can then personalize the system by giving feedback about individual songs, or by adding general guidelines to the station. Feedback about an individual song can be provided using the 5-point rating scale common to many media recommenders, as well as by talking about the song's attributes (e.g., "This song is too mellow, play something more energetic", Figure 3.2). To add general guidelines about the station, the user can tell it to *prefer* or *avoid* descriptive words or phrases (e.g., "Strongly prefer garage rock artists", Figure 3.3, top). Users can also limit the station's search space (e.g., "Never play songs from the 1980's", Figure 3.3, bottom).

AuPair was implemented as an interactive web application, using jQuery and AJAX techniques for real-time feedback in response to user interactions and control over audio playback. We supported recent releases of all major web browsers. A remote web server provided recommendations based on the user's feedback and unobtrusively logged each user interaction via an AJAX call.

AuPair's recommendations were based on data provided by The Echo Nest[2], allowing access to a database of cultural characteristics (e.g., genre, mood, etc.) and acoustic

[2] `http://the.echonest.com`

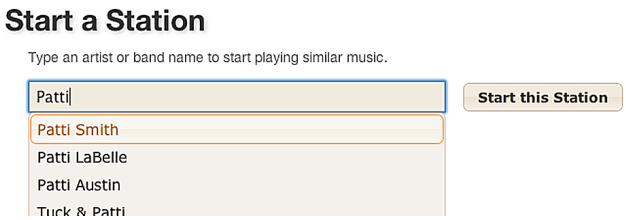


Figure 3.1: Users start a station by specifying an artist they'd like to hear music similar to.

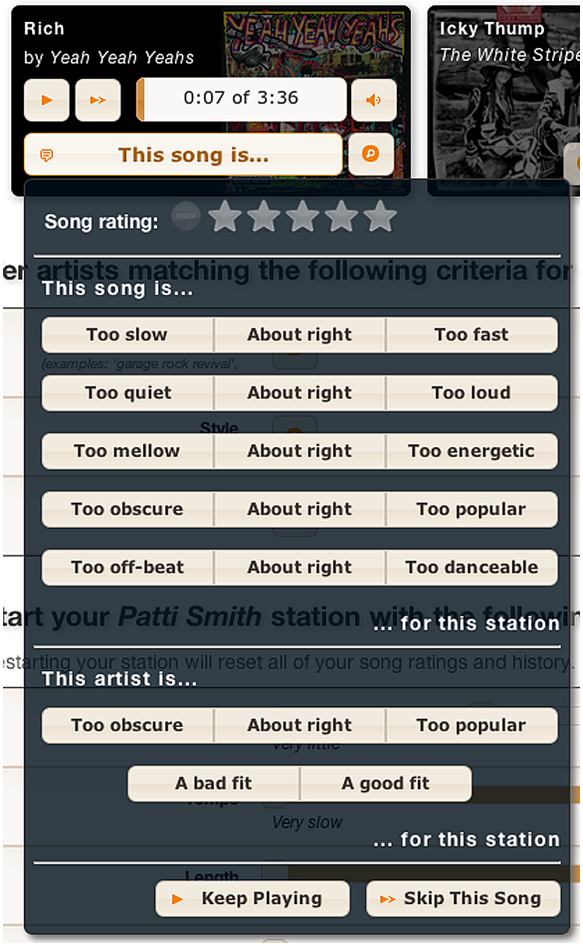


Figure 3.2: Users can personalize their station by saying *why* the current song was a good or bad choice.

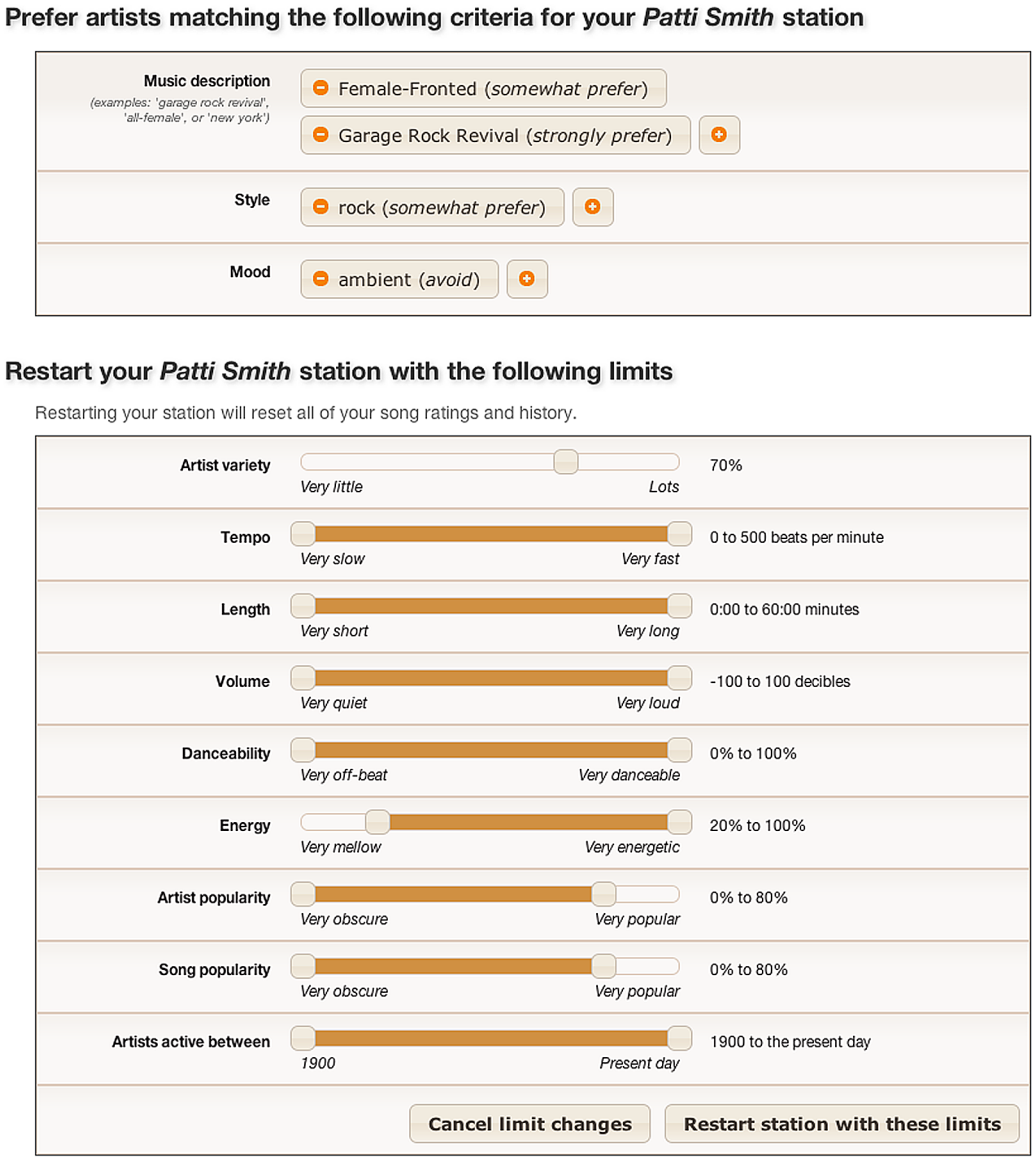


Figure 3.3: Users can place guidelines on the type of music the station should or should not play, via a wide range of criteria.

characteristics (e.g., tempo, loudness, energy, etc.) of the music files in our library. We built AuPair's music library by combining the research team's personal music collections, resulting in a database of more than 36,000 songs from over 5,300 different artists.

The core of our recommendation engine was built upon The Echo Nest API's dynamic playlist feature. Dynamic playlists are put together using machine learning approaches and are "steerable" by end users. This is achieved via an adaptive search algorithm that builds a path (i.e., a playlist) through a collection of similar artists. Artist similarity in AuPair was based on cultural characteristics, such as the terms used to describe the artist's music. The algorithm uses a clustering approach based on a distance metric to group similar artists, and then retrieves appropriate songs. The user can adjust the distance metric (and hence the clustering algorithm) by changing weights on specific terms, causing the search to prefer artists matching these terms. The opposite is also possible—the algorithm can be told to completely avoid undesirable terms. Users can impose a set of limits to exclude particular songs or artists from the search space. User can also query each song or artist to reveal the computer's understanding of its acoustic and cultural characteristics, such as its tempo, genre, "energy", or "danceability".

3.2.2 Participants

Our study was completed by 62 participants, (29 females and 33 males), ranging in age from 18 to 35. Only one of the 62 participants reported prior familiarity with computer science. These participants were recruited from Oregon State University and the local community via email to university students and staff, and fliers posted in public spaces around the city (coffee shops, bulletin boards, etc.). Participants were paid $40 for their time upon completion of the study. Potential participants applied via a website that automatically checked for an HTML5-compliant web browser (applicants using older browsers were shown instructions for upgrading to a more recent browser) to reduce the chance of recruiting participants who lacked reliable Internet access or whose preferred web browser would not be compatible with our AuPair Radio prototype.

3.2.3 Experiment design and procedure

We randomly assigned participants to one of two groups—a *With-scaffolding* treatment group, in which participants received special training about AuPair's recommendation engine, and a *Without-scaffolding* control group. Upon arrival, participants answered a widely used, validated self-efficacy questionnaire (Compeau and Higgins, 1995) to measure their confidence in problem solving with a hypothetical (and unfamiliar) software application.

Both groups then received training about AuPair. The same researcher provided the tutorial to every participant, reading from a script for consistency. To account for differences in participant learning styles, the researcher presented the tutorial interactively, via a digital slideshow interleaved with demonstrations and hands-on participation. For both groups this tutorial included about 15 minutes of instruction about the functionality of AuPair, such as how to create a station, how to stop and restart playback, and other basic usage information. This tutorial was designed to help all participants build a functional mental model of the system.

Following the basic usage tutorial, the With-scaffolding group received an additional 15-minute tutorial about AuPair to help induce a structural mental model of the recommendation engine. This "behind the scenes" training included illustrated examples of how AuPair determines artist similarity, the types of acoustic features the recommender "knows" about, and how it extracts this information from audio files. Researchers systematically selected content for the scaffolding training by examining each possible user interaction with AuPair; for each interaction, the tutorial included an explanation of how the recommender would respond. For instance, every participant was told that the computer will attempt to "play music by similar artists", but the With-scaffolding participants were then taught how *TF-IDF* (term frequency-inverse document frequency, a common measure of word importance in information retrieval (Ramos, 2003)) measurements were used by the system to identify "similar" artists. In another instance, every participant was shown a control for using descriptive words or phrases to steer the system, but only With-scaffolding participants were told where these descriptions came from (traditional sources, like music charts, as well as Internet sources, such as Facebook pages).

After this introduction, each participant answered a set of six multiple-choice comprehension questions in order to establish the fidelity of their mental models. Each question presented a scenario (e.g., "Suppose you want your station to play more music by artists similar to The Beatles"), and then asked which action, from a choice of four, would best align the station's recommendations with the stated goal. Because mental models are inherently "messy, sloppy... and indistinct" (Norman, 1987), we needed to determine if participants were guessing, or if their mental models were sound enough to eliminate some of the incorrect responses. Thus, as a measure of confidence, each question also asked how many of the choices could be eliminated before deciding on a final answer. A seventh question asked participants to rate their overall confidence in understanding the recommender on a 7-point scale.

The entire introductory session (including questionnaires) lasted 30 minutes for Without-scaffolding participants and 45 minutes for With-scaffolding participants. Both groups received the same amount of hands-on interaction with the recommender.

Over the next five days, participants were free to access the web-based system as they pleased. We asked them to use AuPair for at least two hours during this period, and to create at least three different stations. Whenever a participant listened to music via AuPair, it logged usage statistics such as the amount of time they spent personalizing the system, which personalization controls they used, and how frequently these controls were employed.

After five days, participants returned to answer a second set of questions. These included the same self-efficacy and comprehension questionnaires as on Day 1 (participants were not told whether their prior comprehension responses were correct), plus the NASA-TLX survey to measure perceived task load (Hart and Staveland, 1988). We also asked three Likert-scale questions about user's satisfaction with AuPair's recommendations (using a 21-point scale for consistency with the NASA-TLX survey) and the standard Microsoft Desirability Toolkit (Benedek and Miner, 2002) to measure user attitudes toward AuPair.

3.2.4 Data analysis

We used participants' answers to the comprehension questions described earlier to measure mental model fidelity. Each question measured the depth of understanding

for a specific type of end user debugging interaction, and their combination serves as a reasonable proxy for participants' understanding of the entire system. We calculated the fidelity of participant's mental models using the formula $\sum_{i=1}^{6}(correctness_i \cdot confidence_i)$, where *correctness* is either 1 for a correct response or -1 for an incorrect response, and *confidence* is a value between 1 and 4 (representing the number of answers the participant was able to eliminate). These values were summed for each question i to create a participant's comprehension score, ranging from -24 (indicating a participant who was completely confident about each response, but always wrong) to +24 (indicating someone who was completely confident about each response and always correct).

As discussed in Section 2.1, measuring in-the-head constructs such as mental models is challenging—participants may give incomplete answers that do not reflect the entirety of their mental model, and the very act of reflecting on their mental models may cause them to change. Thus, a participant with a perfect mental model score (as described above) does not necessarily hold a perfect mental model—even if several participants all held similar models, we would expect to see some variance in their mental model scores because of the challenges inherent to capturing mental models. However, statistical tests account for variance, so we can perform between-subject tests to determine if one group of subjects tended to hold better mental models than the other, and this is the approach we take throughout this dissertation. While such tests provide evidence that the mental model scores between two groups differed (e.g., a high-fidelity group and a low-fidelity group), we must still refrain from claiming that these scores reveal the *exact* fidelity of participants' mental models.

Mental models evolve as people integrate new observations into their reasoning (Norman, 1987), and previous studies have suggested that participants may adjust their mental models while working with a learning system that is transparent about its decision-making process (Kulesza et al., 2010). Furthermore, constructivist learning theory (Kolb, 1984) places emphasis on knowledge transformation rather than the overall state of knowledge. Hence, we also calculated mental model transformation by taking the difference of participants' two comprehension scores (*day_5_score – day_1_score*). This measures how much each participant's knowledge shifted during the study, with a positive value indicating increasing fidelity, and a negative value suggesting the replacement of high-fidelity models with low-fidelity models.

Metric	Definition
Mental model fidelity	Responses to comprehension questions (sum of correct responses, weighted by confidence).
Perceived mental model fidelity	Response to Likert question "Are you confident all of your statements are accurate?" after participants were asked to enumerate how they think the recommender made decisions.
Mental model transformation	Difference between post-task mental model fidelity and pre-task mental model fidelity.
Personalization interactions	Number of actions a participant used to personalize each playlist (e.g., providing feedback, getting the next recommendation, or viewing a song's features), from the automated log files.
Interaction time	Length of time a participant spent on the task, i.e. listening to and interacting with AuPair.
Cost/benefit	Response to Likert question "Do you feel the effort you put into adjusting the computer was worth the result?"
Satisfaction	Response to Likert question "How satisfied are you with the computer's playlists?"

Table 3.1: Definitions for each metric used in our data analysis.

Table 3.1 lists all of our metrics and their definitions.

3.3 Results

3.3.1 Feasibility (RQ3.1)

3.3.1.1 Effectiveness of scaffolding

Understanding how machine learning systems work is not trivial—even designers and builders of intelligent systems may have considerable difficulty (Kapoor et al., 2010).

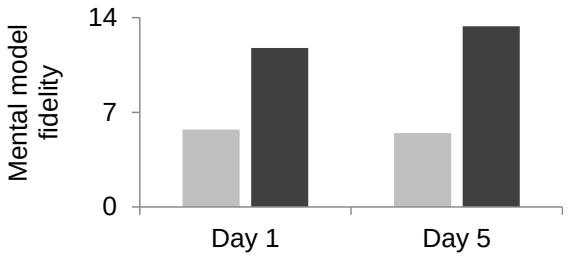


Figure 3.4: With-scaffolding participants (dark) held sounder mental models than without-scaffolding participants (light), both immediately following the tutorial, and five days later.

Our first research question (RQ3.1) considers the feasibility of inducing a high-fidelity mental model of an algorithm's reasoning process in end users—if participants fail to learn how the recommender works given a human tutor in a focused environment, it seems unreasonable to expect them to easily learn it on their own.

We tested for a difference in mental model fidelity (as measured by comprehension scores weighted by confidence) between the With-scaffolding group and the Without-scaffolding group. The With-scaffolding group had significantly higher scores than the Without-scaffolding group, both before and after the experiment task (Day 1: Welch's t-test, $t(54) = -3.03$, $p = .004$) (Day 5: Welch's t-test, $t(60) = -3.77$, $p < .001$). To ensure these differences were not primarily the result of differing levels of confidence, we performed the same test without weighting the comprehension scores by confidence, finding nearly identical results (Day 1: Welch's t-test, $t(55) = -3.09$, $p = .003$) (Day 5: Welch's t-test, $t(59) = -3.55$, $p < .001$). Neither group's mean comprehension score changed significantly during the 5-day study (Figure 3.4).

Participants also showed differences in their *perceived* mental model fidelity, at least at first. On Day 1, the Without-scaffolding group was significantly less certain that they accurately understood how the system selected songs and responded to feedback (mean score of 4.5 out of 7) than the With-scaffolding group (mean score of 5.6 out of 7) (Welch's t-test, $t(58) = -2.51$, $p = .015$). By Day 5, however, the Without-scaffolding group's perceived mental model fidelity responses had risen to a mean of 5.25, with no evidence of statistical difference against the With-scaffolding group (with a mean of 5.3).

3.3.1.2 Discussion

These results provide insights into four aspects of the practicality of end users comprehending and debugging the reasoning of a machine learning system.

First, even a short 15-minute scaffolding tutorial effectively taught participants how the recommender reasoned. With-scaffolding participants were significantly more likely to correctly and confidently answer the comprehension questions. This in turn suggests that the With-scaffolding participants should be better equipped to debug the recommender's reasoning than the Without-scaffolding participants, a point we investigate in RQ3.2.

Second, mental model fidelity did not significantly improve during the five days participants interacted with AuPair on their own—simply *using* the system did not significantly help participants develop more accurate mental models about its reasoning. This is in contrast to recent work in interactive machine learning, which has found that for some systems (e.g., gesture recognition frameworks), repeated use taught people the most salient aspects of how the system worked (Fiebrink et al., 2011).

Third, the fidelity of participants' mental models largely persisted for the duration of the study. This appeared to be the case for both the Without-scaffolding and With-scaffolding groups, with neither groups' comprehension scores significantly changing between Day 1 and Day 5. This bodes well for end users retaining and recalling sound models initially learned about a machine learning system.

Fourth, however, is the issue of initially building inaccurate, low-fidelity models: once low-fidelity models were built, they were unlikely to improve. Even though the Without-scaffolding group formed low-fidelity mental models, their confidence in these mental models increased during the course of the experiment, suggesting that they had convinced themselves they were, in fact, correct. Making *in situ* explanations available on an ongoing basis, such as in (Kulesza et al., 2010; Talbot et al., 2009; Herlocker et al., 2000), may be a way to address this issue.

Together, these findings provide evidence that furnishing end users with a brief explanation on the structure of a machine learning system's reasoning, such as the attributes used, how such attributes are collected, and the decision-making procedure employed, can significantly improve their mental model's fidelity.

3.3.2 Personalization (RQ3.2)

A recommender's effectiveness is in the eye of the beholder. Personalized recommendations cannot have a "gold standard" to measure accuracy—only the end users themselves can judge how well an system's recommendations match their personal expectations. Hence, for our second research question (RQ3.2), we turned to a pair of more appropriate measures to explore the effects of mental model fidelity on personalization—cost/benefit and participant satisfaction.

3.3.2.1 Cost/Benefit

In theory, a high-fidelity mental model enables a person to reason effectively about their best course of action in a given situation (Johnson-Laird, 1983). Thus, we expected participants with high-fidelity mental models (the With-scaffolding participants, according to the RQ3.1 results) to personalize AuPair more effectively than those with low-fidelity mental models. For example, knowing that the recommender could be steered more effectively by using unique, highly specific words (e.g., "Merseybeat") rather than broad, common descriptors (e.g., "oldies") should have helped such participants personalize the recommender's reasoning more effectively than participants who did not understand this.

Surprisingly, when using participants' perceptions of cost/benefit as a surrogate for effectiveness, the soundness of participants' mental models showed little impact on this measure of personalization effectiveness. Mental model *transformation*, however, was tied with cost/benefit: participants who most improved the fidelity of their mental models reported that the effort of personalizing their radio station was significantly more worthwhile than participants whose mental models improved less, or not at all (Table 3.2, row 1 and Figure 3.5, top left).

Participants' opinions of effectiveness were confirmed by their interactions to adjust or assess AuPair's recommendations (e.g., providing feedback, getting the next recommendation, or viewing a song's features). The count of these personalization interactions was significantly correlated with the improvement in mental model fidelity for With-scaffolding participants, while no such correlation existed among Without-scaffolding participants (Table 3.2, rows 2 and 3, and Figure 3.5, top right). Improvements to a

participant's mental model, then, may have had a positive impact on their ability to personalize the system, whereas small changes to an initially incorrect model did not serve the Without-scaffolding participants as well.

Further, participants who most improved the fidelity of their mental models spent significantly less time on their interactions than other participants (Table 3.2, row 4, and Figure 3.5, bottom left). In light of the increases in perceived cost/benefit and personalization interactions, this suggests positive mental model transformations were linked to more efficient personalization.

An alternative explanation of the above results is that personalization interactions were responsible for participants' mental model transformations, rather than the other way around. Recall, however, that the Without-scaffolding group showed no correlation between the number of personalization interactions and their mental model scores (Table 3.2, row 3). Thus, the evidence suggests that it was the *in situ* enhancement of already relatively high-fidelity models that was linked to improved attitudes toward personalization.

3.3.2.2 Satisfaction

Our second measure of personalization effectiveness was participants' satisfaction with AuPair's resulting recommendations. To measure this, we asked participants (via a Likert scale) "How satisfied are you with the computer's playlists?" at the end of the study.

As with the cost/benefit results, neither treatment nor mental model fidelity was predictive of participant satisfaction (Table 3.2, rows 5 and 6). However, here again, transformation of mental models appeared to matter—mental model transformation was marginally predictive of how satisfied participants felt with AuPair's playlists (Table 3.2, row 7). For example, the participant whose mental model's fidelity decreased the most expressed dissatisfaction and a feeling of being unable to control the computer:

> *"The idea is great to be able to 'set my preferences', but if the computer continues to play what I would call BAD musical choices—I'd prefer the predictability of using Pandora."*

Conversely, one of the participants whose mental model most increased in fidelity expressed a feeling of being more in control:

	Metric	Statistical test	Result	Figure
1	Mental model transformation vs. cost/benefit	Linear regression	$R^2 = .07$, $F(1, 60) = 4.37$, $p = .041$	Figure 3.5 (top left)
2	Mental model transformation (With-scaffolding) vs. personalization interactions	Pearson correlation	$r(28) = .39$, $p = .031$,	Figure 3.5 (top right)
3	Mental model transformation (Without-scaffolding) vs. personalization interactions	Pearson correlation	$r(30) = .01$, $p = .952$	
4	Mental model transformation vs. interaction time	Pearson correlation	$r(60) = -.27$, $p = .032$	Figure 3.5 (bottom left)
5	Satisfaction between With-scaffolding/ Without-scaffolding groups	Welch's t-test	$t(60) = 1.53$, $p = .129$	
6	Satisfaction vs. mental model fidelity	Linear regression	$R^2 = .02$, $F(1, 60) = 1.23$, $p = .272$	
7	Satisfaction vs. mental model transformation	Linear regression	$F(1, 60) = 3.89$, $R^2 = .06$, $p = .053$	
8	Satisfaction vs. cost/benefit	Pearson correlation	$r(60) = .73$, $p < .001$	Figure 3.5 (bottom right)
9	Satisfaction vs. personalization interactions	Pearson correlation	$r(60) = -.13$, $p = .293$	

Table 3.2: Positive mental model transformations were consistently associated with better benefits, lower costs, and improved satisfaction (significant results shaded). Definitions for each metric are listed in Table 3.1.

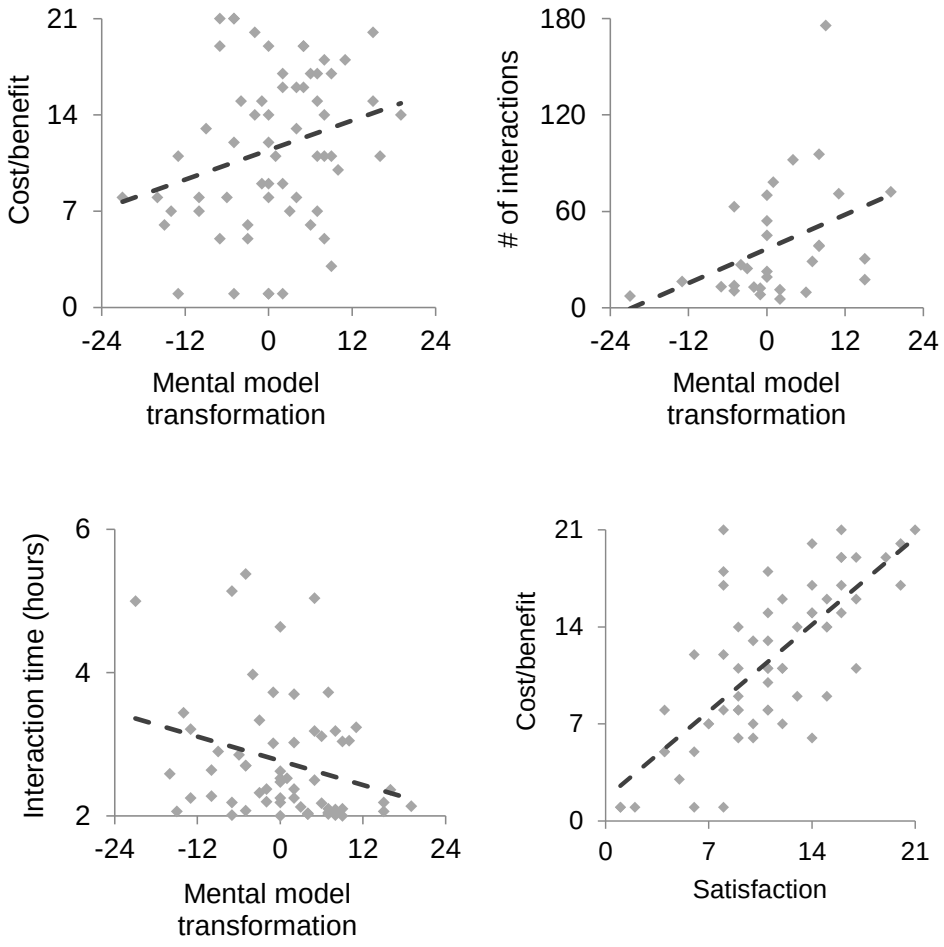


Figure 3.5: Scatterplots of raw data for each significant result from Table 3.2. Definitions for axis measurements are listed in Table 3.1.

> *"I like the idea of having more control to shape the station. Controls made sense and were easy to use. The user has a lot of options to tune the station."*

Perceived cost/benefit from personalizing the recommender was also significantly correlated with participant satisfaction (Table 3.2, row 8, and Figure 3.5, bottom right)—further evidence that satisfaction was indicative of an increased ability to personalize the learning system's reasoning. To ensure that participant satisfaction was not simply a result of time and effort invested, we tested for a relationship between reported satisfaction and the number of personalization interactions each participant performed, but found no evidence of a correlation (Table 3.2, row 9).

3.3.2.3 Discussion

An additional factor may have affected participant satisfaction: whether or not AuPair Radio *could* play the music they were hoping to hear. Our music database held songs by just over 5,300 artists—pandora.com, by comparison, has over 80,000 different artists.[3] Participant satisfaction may have been confounded by the fact that some participants hoped their stations would play music that was unavailable to AuPair. As one participant commented:

> *"The songs played weren't what I was looking for, the selection was poor. The system itself was excellent, but I need more music."*

Despite this potential factor, the confluence of several metrics (cost/benefit, personalization interactions, interaction time, and satisfaction) suggests that transformations in mental model fidelity translated to an improved ability to personalize the recommender's reasoning, resulting in more satisfaction with AuPair's recommendations. Because our evidence suggests mental model transformations (which occurred *during* the study) helped participants personalize the system more efficiently and effectively, continuing to provide explanations of a learning system's reasoning while end users interact with it may help to increase their ultimate satisfaction with the system's predictions or recommendations. Such on-line explanations, however, were not investigated by the

[3] Pandora Media, Inc. Initial Public Offering Form S-1 (2011).

current study; we focused our exploration on the impact of explanations prior to (rather than during) user interaction with a machine learning system.

One potential explanation of why we found no evidence that end-of-study mental model fidelity was predictive of personalization ability could be that the information presented to the With-scaffolding tutorial participants was not helpful for *personalizing* the recommender's reasoning; it may have only been helpful for *understanding* the recommender. Instead, the most effective participants may have built upon this initial understanding to quickly learn how to efficiently personalize AuPair while interacting with the system. However, this alternative explanation is weakened by the fact that the prototype was not transparent about how it made its decisions; the only time when participants were presented with explanations of AuPair's reasoning occurred during the With-scaffolding tutorial.

3.3.3 Confidence (RQ3.3)

Presenting a complex system to unsuspecting users could overwhelm them. We are particularly concerned with peoples' willingness to personalize (or otherwise adjust) the reasoning of machine learning systems—some people (especially those with low computer self-efficacy) may perceive a risk that their debugging is more likely to harm the system's reasoning than to improve it. Similarly, computer anxiety (a "degree of fear and apprehension felt by individuals when they consider the utilisation, or actual use, of computer technology" (Bozionelos, 2001)) is known to negatively impact how (and how well) people use technology, and is negatively correlated with computer self-efficacy (Wilfong, 2006).

Surprisingly, we found the opposite to be true—teaching participants about AuPair's reasoning may have helped *increase* their computer self-efficacy. As Table 3.3 shows, almost three-quarters of the With-scaffolding participants experienced an increase in their computer self-efficacy between Day 1 and Day 5. Without-scaffolding participants, conversely, were as likely to see their computer self-efficacy decrease as to increase. A χ^2 comparison showed that With-scaffolding participants were significantly more likely than a uniform distribution (in which only half would increase their self-efficacy) to increase their computer self-efficacy ($\chi^2(1, N = 62) = 6.53$, $p = .011$). This suggests that exposure to the internal workings of machine learning systems may have helped to

	Self-efficacy...		
	Did improve	**Did not improve**	**Average change**
Without-scaffolding	16	16	3.29%
With-scaffolding	22	8	5.90%

Table 3.3: Participants in the With-scaffolding group were likely to end the experiment with higher computer self-efficacy than when they began.

allay, rather than to increase, participants' perceived risk of making their personalized learning system worse.

As further evidence that it was understanding how the system worked (rather than simply a byproduct of using it) that influenced participants' computer self-efficacy, participants' perceived mental model fidelity was significantly correlated with their computer self-efficacy at the end of the study (Pearson correlation, $r(60) = .44$, $p < .001$). Additionally, there was no evidence of a correlation between the number of personalization interactions participants made and their self-efficacy at the end of the study (Pearson correlation, $r(60) = .13$, $p = .286$); participants did not appear to grow more confident by simply interacting with the system. Thus, participants who at least thought they understood the nuances of AuPair's reasoning scored higher on the computer self-efficacy questionnaire than those who expressed little confidence in their knowledge of the recommender's logic.

3.3.3.1 Discussion

We hope further research will shed additional light on this preliminary link between learning how a machine learning system reasons, and increasing levels of computer self-efficacy (and, by association, decreasing levels of computer anxiety). Challenging tasks, when successfully accomplished, have been found to have a significantly larger impact on self-efficacy than overcoming small obstacles (Bandura, 1977). Personalizing a machine learning system seems exactly the sort of difficult computer task that, successfully carried out, may make people say, "If I could do that, surely I can do this other thing...", thereby reducing the obstacles of risk and anxiety toward future computer interactions.

3.3.4 User experience (RQ3.4)

For our final research question, we looked at the potential effects of mental model fidelity on perceptions of experience, such as cognitive demands and emotional responses.

3.3.4.1 Cognitive demands

Prior work has found that explaining concrete decisions of a learning system's reasoning to end users *in situ* created an increase in participants' frustration with, and mental demand of, correcting the system's mistakes (measured via the NASA-TLX questionnaire) (Kulesza et al., 2010). We suspected that end users might experience similar effects when presented with prior structural knowledge. However, the With-scaffolding participants showed no significant difference to Without-scaffolding participants' TLX scores. While acquiring a high-fidelity mental model undoubtedly requires mental effort on the part of end users, we encouragingly found no evidence that this was any greater than the mental effort required to interact with the learning system while lacking a clear understanding of its underpinnings. This suggests that end users' experience with learning systems does not necessarily suffer when they are exposed to more knowledge of how the system works.

3.3.4.2 Emotional responses

We used the Microsoft Desirability Toolkit (Benedek and Miner, 2002) to investigate participants' user experience with the AuPair music recommender. Participants were given a list of 118 adjectives and asked to underline each one they felt was applicable to their interactions with AuPair.

The Internet General Inquirer (a tool that associates participants' words with either positive or negative connotations, based on the content analysis framework proposed in (Stone, 1968)) revealed that With-scaffolding participants employed slightly more positive descriptions of AuPair than the Without-scaffolding group (54.9% vs. 49.6%) and fewer negative descriptions (9.9% vs. 12.0%). While not statistically significant between groups, these numbers suggest that the With-scaffolding participants (with their higher-fidelity mental models) may have viewed the overall experience of interacting with AuPair in a more positive light than Without-scaffolding participants.

Figure 3.6: Tag cloud of negative descriptive terms for AuPair. Without-scaffolding participants found the system "overwhelming" and "complex" (top), whereas the With-scaffolding group (bottom) viewed it as "simplistic".

Participants' descriptions revealed a subtler picture of the difficulties they faced. Word clouds—in which a word's frequency is indicated by its size—of the negative descriptions show that the With-scaffolding group's complaints may have stemmed more from difficulties using the system than difficulties understanding it; these participants were apt to complain the system was "simplistic", "annoying", and "frustrating" (Figure 3.6, bottom), while the Without-scaffolding group appeared to have trouble even understanding the impact of their debugging interactions, citing the system as "confusing", "complex", "overwhelming", and "ineffective" (Figure 3.6, top).

Participants' choices of positive descriptions provide further evidence the With-scaffolding participants' mental models contributed positively to interacting with the system (Figure 3.7). The phrase "easy to use" dominated their responses, alongside "innovative" and "accessible". In contrast, the Without-scaffolding participants focused on the visual appearance of the system, with words like "clean" and "appealing". Participants with a deeper understanding of the system may have placed more emphasis on the interaction experience than aesthetics.

3.3.4.3 Discussion

Numerous benefits are associated with high-fidelity structural mental models, and in the case of this machine learning system, it appears possible to gain these without impairing

Figure 3.7: Tag cloud of positive descriptive terms for AuPair. Without-scaffolding participants (top) focused on visual appearance more than With-scaffolding participants (bottom).

the user experience. This is encouraging for the feasibility of end-user personalization of recommendation systems (and other types of learning systems), especially when the user associates a benefit with personalizing the system's reasoning.

3.4 Conclusion

This chapter provides the first empirical exploration of how mental models impact end users' attempts to personalize a machine learning system. By scaffolding structural mental models for half of our study's participants, we learned that:

- Despite the complexity inherent with machine learning, With-scaffolding participants quickly built high-fidelity mental models of how one such system (a music recommender) operates "behind the scenes"—something the Without-scaffolding participants failed to accomplish over five days of system usage.

- The participants' mental model transformations—from lower to higher fidelity—was predictive of their ultimate satisfaction with the learning system's output. Participants with the largest transformations were able to efficiently personalize their recommenders' reasoning, aligning it with their own reasoning better and faster than other participants. These same participants were also likely to perceive a greater benefit from their personalization efforts.
- Participants presented with structural knowledge of the learning system's reasoning were significantly more likely to increase their computer self-efficacy, which is known to correlate with reduced computer anxiety and increased persistence when tackling complex computer tasks.
- Participants who were presented with structural knowledge showed no evidence of feeling overwhelmed by this additional information and viewed interacting with the learning system in a positive light, while participants holding only functional mental models more frequently described their personalization experience in negative terms, such as "confusing" and "complex".

This work demonstrates the value and practicality of providing end users with structural knowledge of their machine learning systems' reasoning. Our results suggest that if users acquire better mental models of a learning system (particularly while interacting with it), they will be better able to personalize that system. However, the *if* may be a stumbling block. In this study, human-led instruction helped our With-scaffolding participants to build higher-fidelity mental models than Without-scaffolding participants, but this is not a viable solution for users of machine learning systems. Instead, the learning system itself will need to help users build high-fidelity mental models, and it is this challenge we turn to in the next chapter.

Chapter 4: How explanations can impact mental model fidelity

4.1 Introduction

The previous chapter found that it is feasible to help end users acquire high-fidelity mental models of a machine learning system, but because it relied upon human instruction to induce these models, an important question was left unanswered: how should machine learning systems explain themselves to users? The predominant approach in commercial systems is to "keep it simple" (e.g., the music recommender Pandora.com currently describes its song recommendations via a single sentence; the junk mail classifier in Gmail explains each spam classification using a set of just eight high-level reasons[1]). Such simplicity, however, may prevent users from understanding how the system makes decisions, and as we saw in Chapter 3, this may in turn prevent users from successfully personalizing their learning systems.

In this chapter we raise the question of whether simplicity is the right attribute to prioritize when designing a machine learning system's explanations. A different possibility is to prioritize explanation *fidelity*—how closely the explanation matches the underlying system—as this may then help end users build high-fidelity mental models of the learning system. However, information comes at the price of attention—a user's time (and interest) is finite, so high-fidelity explanations may discourage users from attending to their contents.

To investigate how machine learning systems should explain themselves to their users, we performed a qualitative study to separately consider two dimensions of explanation fidelity: *soundness* (how truthful each element in an explanation is with respect to the underlying system) and *completeness* (the extent to which an explanation describes all of the underlying system). We then investigated how varying soundness and completeness (as in Figure 4.1) impacted users' mental models of a music-recommending machine learning system, what types of information (based on Lim and Dey's classification of intelligibility types) were most helpful in the explanations, how explanation fidelity

[1] Reasons for Gmail's spam classifications: `http://bit.ly/13ODtwl`

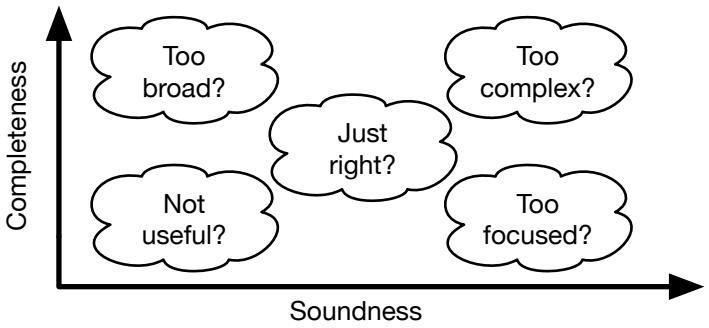


Figure 4.1: Our problem space: How sound and complete do explanations need to be to help end users build high-fidelity mental models?

impacted users' perceptions of the costs and benefits of attending to these explanations, and users' trust in the explanations' veracity. Our research questions were:

RQ4.1: How do the *soundness* and *completeness* of explanations impact end users' mental models?

RQ4.2: Which machine learning *intelligibility types* are most helpful for the development of users' mental models?

RQ4.3: What *obstacles* do end users encounter when building mental models of a learning system's reasoning?

RQ4.4: How do users' perceived *costs and benefits* of attending to explanations change as explanation fidelity increases?

RQ4.5: How does user *trust* change as explanation fidelity increases?

4.2 Explanation soundness and completeness

We tease apart *soundness* and *completeness* because machine learning system designers can make choices independently in each as to the fidelity of their learning systems' explanations. The terms soundness and completeness are borrowed from the field of formal logic, in which a deductive system is *sound* if all of the statements it can create

evaluate to true, and *complete* if its compositional rules allow it to generate every true statement. We apply these terms to explanations in an analogous manner:

Soundness ("nothing but the truth"): the extent to which *each component of an explanation's content* is truthful in describing the underlying system.

Completeness ("the whole truth"): the extent to which *all of the underlying system* is described by the explanation.

For example, a learning system that explains its reasoning with a simpler model than it actually uses (e.g., a set of rules instead of additive feature weights) is reducing soundness, whereas a system that explains only some of its reasoning (e.g., only a subset of a user neighborhood) is reducing completeness.

Together, soundness and completeness let us systematically explore explanation fidelity by varying both dimensions independently of one another. One hypothesis is that more information in an explanation will help users build better mental models. However, very complete or complex explanations require more attention to process, which disincentivizes users to build accurate mental models. Rosson et al., for example, found that the Minimalist explanation model (Carroll and Rosson, 1987)—which minimizes passive learning tasks, such as reading long explanations, favoring instead short explanations coupled with try-it-out exploration—helped programmers understand Smalltalk programs up to two orders of magnitude faster than traditional instruction techniques (Rosson et al., 1990).

4.3 Methodology

To investigate our research questions, we presented 17 participants with up to 8 music recommendations made by a functional prototype. For each recommendation, the participant was given several types of explanations for why the system choose that song and was then asked why they thought the system made the recommendation.

4.3.1 Prototype recommender system

We developed a prototype music recommender to make personalized song recommendations for each participant. Our prototype used a hybrid recommendation approach,

as such approaches have been shown to out-perform more traditional types of recommenders (Su and Khoshgoftaar, 2009) and also provide more "moving parts" to explain. Specifically, our prototype employed user-based collaborative filtering to find artists and a content-based approach for selecting songs by those artists.

To train our recommender, we collected the listening habits of about 200,000 Last.fm listeners between July 2011 and July 2012 via the Last.fm API[2]. We identified the 50 most-played artists for each of these listeners during this time period, and then used the Mahout framework[3] to build a k-nearest-neighborhood (with $k = 15$), where distance between Last.fm users was based on overlap in the artists they listened to (calculated via the log-likelihood metric (Dunning, 1993)).

Prior to the study, we asked each participant to imagine a situation where they would want a playlist of music, and to tell us five artists representative of the type of music they would like the playlist to include. Our prototype took these artists and, using the technique described above, recommended 20 artists for the given participant (Figure 4.2, top). To select specific songs, our prototype used a bagged decision tree based on Weka's J48 implementation (Hall et al., 2009) (the bagging ensemble consisted of 100 decision trees). This classifier was independently trained for each participant using a set of positive training instances (the top 1,500 songs played by Last.fm listeners in the participant's user neighborhood) and a set of negative training instances (the top 1,500 songs played by Last.fm listeners who did *not* listen to any artists that neighborhood members listened to). This resulted in a classifier able to predict whether a given user would or would not like a particular song, along with a certainty score (Figure 4.2, bottom). The song features (recall that a machine learning *feature* is a piece of information a classifier can use to discriminate between output classes) came from The Echonest's[4] database, which includes information such as a song's tempo, energy, and key.

To determine which songs to recommend to a participant, our prototype collected the 25 most popular songs by each recommended artist, resulting in a a 500 song set per participant. We used these songs' feature vectors as input to our classifier, which predicted whether or not the participant would like each song. The positive results were

[2] http://www.last.fm/api
[3] http://mahout.apache.org
[4] http://developer.echonest.com

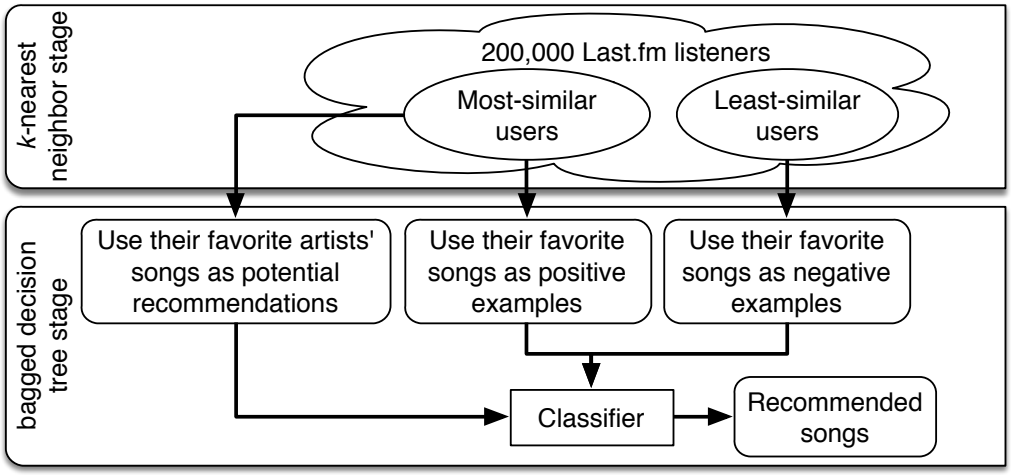


Figure 4.2: Our prototype used a *k*-nearest neighbor stage to identify similar and dissimilar users (top), and a bagged decision tree stage to predict which songs the participant would most enjoy (bottom).

sorted by decreasing certainty, with the top eight used as song recommendations for the participant.

4.3.2 Treatments and explanations

We explored four experiment conditions, which are shown in Table 4.1: HH (high-soundness, high-completeness), MM (medium-soundness, medium-completeness), HSLC (high-soundness, low-completeness), and LSHC (low-soundness, high-completeness). Figure 4.1 visualizes this design space, with HH in the top right, HSLC in the bottom right, LSHC in the top left, and MM in the middle. We used multiple conditions to gather data on a variety of explanation configurations, but restricted ourselves to these four (as opposed to different combinations of soundness and completeness, such as low-soundness/low-completeness) because we felt the combination of three treatments involving extremely high or low soundness and/or completeness, plus one treatment with moderate soundness and completeness, would have the best chance to identify differences in the impact of varying levels of explanation fidelity.

	Treatment	HH	MM	HSLC	LSHC
	Relative soundness	High	Medium	High	Low
	Relative completeness	High	Medium	Low	High
Intelligibility types	Why (song)	Bagged decision tree	Decision tree	Bagged decision tree	Decision stump
	Why (artist)	Nearest neighbor ($k = 15$)	Nearest neighbor ($k = 10$)	Nearest neighbor ($k = 5$)	Nearest neighbor ($k = 15$)
	Certainty	Yes	Yes	No	Yes
	Model	Yes	No	No	Yes
	Input	Yes	Yes	Yes	Yes

Table 4.1: The "Why (song)" intelligibility type was available in all treatments, but its soundness varied. The other intelligibility types were used to vary completeness.

To objectively manipulate completeness, our treatments used a varying number of the *intelligibility types* identified by Lim and Dey (Lim and Dey, 2009): *inputs* (features the system is aware of), *model* (an overview of the learning system's decision making process), *why* (the learning system's reasons for a specific decision), and *certainty* (the system's confidence in a specific decision). We also increased completeness by exposing more information in the *why* (artist) intelligibility type. All treatments included explanations of the song selection process (Figure 4.3), five members of the user's "neighborhood" of similar Last.fm listeners (Figure 4.4), and the features the recommender could use (Figure 4.5). The treatments with more completeness (MM, HH, and LSHC) added the *certainty* intelligibility type (Figure 4.3, bottom left) and showed 10 members of the participant's user neighborhood. The high-completeness treatments (HH and LSHC) also added a high-level description of the recommender's algorithm (the *model* intelligibility type, Figure 4.6) and showed all 15 members of the participant's user neighborhood.

To objectively manipulate soundness, our treatments used a range of simplified models of the recommender's reasons for each song selection. The explanation used in the high-soundness treatments (HH and HSLC) described the bagged decision tree (the

actual algorithm used to produce the playlist). For the medium-soundness treatment (MM), we trained a simpler model (a single J48 decision tree) using the bagged classifier's predicted labels for all of the training instances, and explained this derived model (a variation of the technique in (Craven and Shavlik, 1997)). For the low-soundness treatment (LSHC), we used the same approach to train an even simpler model (a one-feature decision tree, or *decision stump*) to explain (Figure 4.3, bottom right). Because the low-soundness model only explained one highly discriminative feature, we considered it a functional analog for contemporary machine learning system explanations (e.g., a movie recommender that explains its selections by their genres, or a product recommender that explains it is recommending product *foo* because you previously purchased product *bar*).

4.3.3 Participants and study task

We recruited 17 participants (10 females, 7 males) from the local community via flyers and announcements to university mailing lists. Participants' ages ranged from 19 to 34, none had a background in computer science, and each was randomly assigned to one of the four treatments.

During the study, participants listened to their recommended playlist while a researcher provided participants with the paper explanations described in Section 4.3.2. After each song, a researcher asked the participant why they thought it had been recommended. At the end of the study we measured participants' mental models via a combination of short-answer and Likert scale questions. Each session was videotaped and later transcribed.

4.3.4 Data analysis

To qualitatively analyze participants' data, we used grounded theory methods (Corbin and Strauss, 1990) to develop a code set that identified which aspects of the learning system participants understood, and which aspects caused participants to request more information. The resulting code set is presented in Table 4.2.

We transcribed participant utterances during each song and applied the codes to these utterances (each code could be applied, at most, once per song). Two researchers

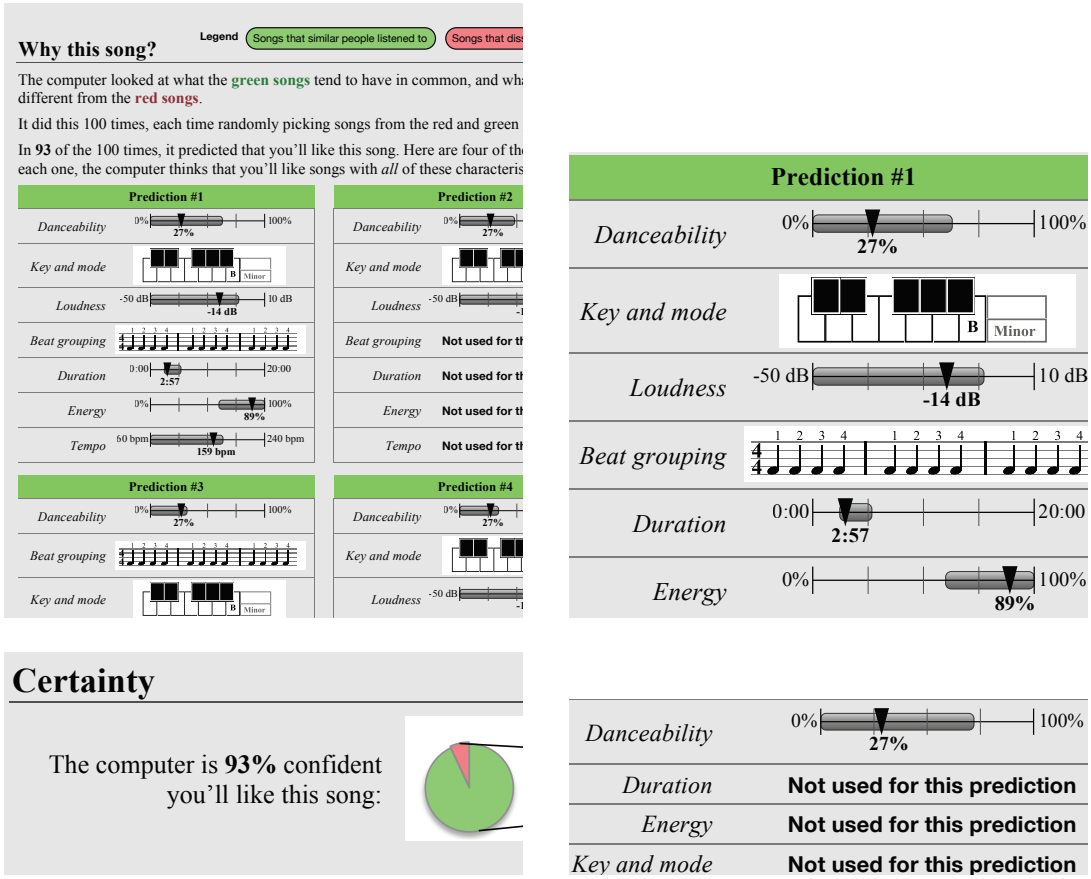


Figure 4.3: Excerpts from the Why this Song explanation (*why* intelligibility type). (Top left): The high-soundness sheet showed a random sample of four decision trees from the bagging ensemble. (Top right): Each tree was represented as a set of ordered features with allowed ranges of values. The medium-soundness explanation was similar, but only showed one derived decision tree that approximated the bagging ensemble's reasoning. (Bottom right): The low-soundness sheet was also similar, but only showed one derived decision stump (a single-featured decision tree). (Bottom left): For the HH, LSHC, and MM treatments, this sheet also included the *certainty* intelligibility type.

The following **15 people** listened to more of the bands you listed than anyone else (out of 200,000 different listeners). The computer used these bands as a starting point to find songs you might like.

Person #1:	*Person #6:*	*Person #11:*
• William Fitzsimmons (#1)	• Jimmy Page & The Black Crowes (#1)	• The Verve Pipe (#1)
• City & Colour (#2)	• The Black Crowes (#2)	• Pearl Jam (#2)
• The Kooks (#3)	• Vitamin String Quartet (#3)	• Third Eye Blind (#3)
• Dream Theater (#4)	• Chris & Rich Robinson (#4)	• Alice in Chains (#4)
• **Foo Fighters (#9)**	• **Pearl Jam (#12)**	• **Foo Fighters (#18)**
• **Three Doors Down (#34)**		

Figure 4.4: Excerpt from Why this Artist (*why* intelligibility type), which showed the artists selected by their user neighborhood. All participants received this explanation, but with different neighborhood sizes (see Table 4.1).

The computer knows the following details about each song:

Danceability	How danceable the computer thinks this song is (as a percentage)
Duration	How long the song is (in minutes)
Energy	How energetic the song is (as a percentage)

Figure 4.5: Excerpt from What the Computer Knows (*input* intelligibility type), which showed a comprehensive list of features that the recommender used. All participants received this explanation.

independently coded a small portion of the transcripts and then discussed areas of disagreement. Once the researchers agreed on the coding of these transcripts, they independently coded two complete transcripts (12% of the total data)—their agreement, as calculated by the Jaccard index (the intersection of all applied codes over the union of all applied codes), was 83%. Given this acceptable level of agreement, a single researcher coded the remaining transcripts and post-study questionnaires.

In this analysis, participants' mental model scores were the number of correct statements, minus the number of incorrect statements, participants made during the experiment and on the post-study questionnaire. For convenience, we translated the raw score (which could be negative if participants made more incorrect than correct

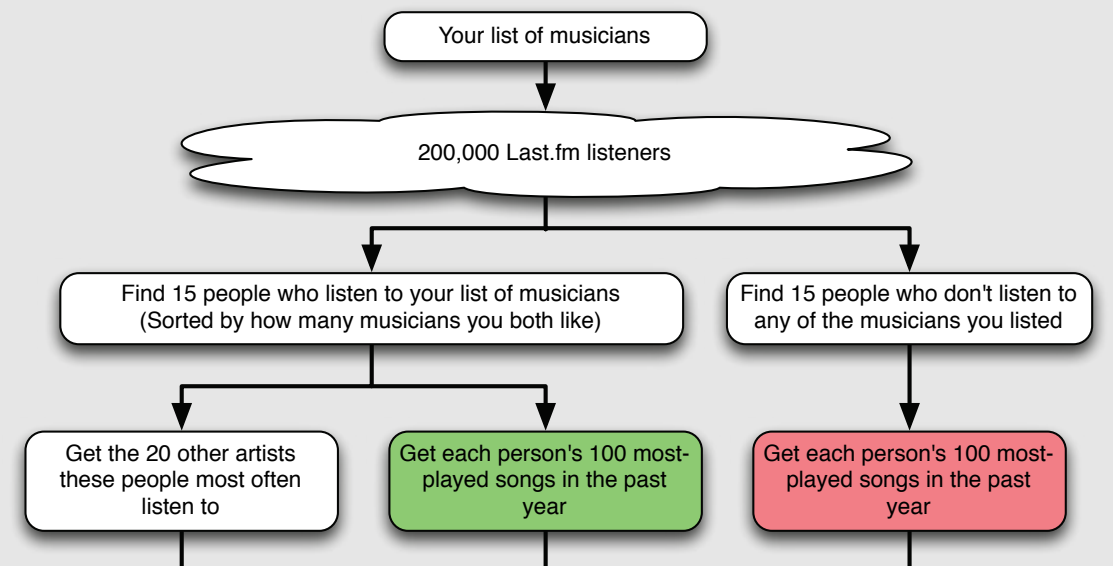


Figure 4.6: Excerpt from How it All Works Together (*model* intelligibility type), which showed how the participants' artists list was used to make song recommendations. Positive and negative training sets were color-coded throughout the flow-chart. Only HH and LSHC participants received this explanation.

statements) to a 0-to-10 scale. Table 4.2 shows which types of participant verbalizations and questionnaire responses were considered correct vs. incorrect. Participants' verbalizations during the study and post-study questionnaire responses were weighted equally when computing mental model scores.

4.4 Results

4.4.1 Soundness, completeness, and intelligibility types (RQ4.1 and RQ4.2)

As Figure 4.7 shows, HH participants achieved three of the top four scores. In contrast, all but one of the participants in the other treatments clustered around lower scores. This surprised us because we had expected the HH treatment may overload participants to the point where they would not attend to so much complex information. Instead, we

Category	Code	Participant discussed/said…
Correct: the participant correctly discussed an aspect of the recommender	Valid artist process	…the artist was chosen via collaborative filtering
	Valid song feature	…specific song features used by the recommender
	Valid song process	…a combination of features were responsible for the recommendation
Incorrect: the participant incorrectly discussed an aspect of the recommender	Invalid feature	…specific features not used by the recommender
	Invalid process	…the computer's reasoning involved a single path through a decision tree or another incorrect description of the artist/song selection process.
Knowledge gaps: the participant expressed uncertainty about their knowledge of the recommender	Don't know	…not knowing how the recommender works
	Uncertain	…uncertainty regarding their answer of how the recommender works
	More explanation details	…needing more details about the explanations
	More recommender details	…needing more details about the recommender

Table 4.2: Code set used to assess participants' mental models.

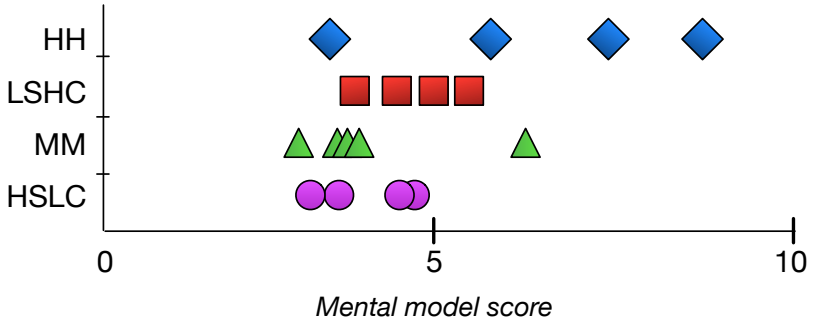


Figure 4.7: Participants' mental model fidelity scores. Each mark is one participant's score. (Note: MM had one more participant than the others.) The highest scores were mostly those of HH participants.

expected the MM treatment to be a "sweet spot" in the trade-off between informativeness and simplicity. Most of the MM participants, however, clustered around the *lowest* mental model scores.

Further, HH participants' mental model scores were consistently high across features and processes, as Figure 4.8's results from the post-task questionnaire show. In fact, HH participants were the only ones to correctly describe the song selection process (third column of Figure 4.8, coded as per Table 4.2), and only one HH participant made any incorrect post-task observations at all (right half of Figure 4.8). (Note from Table 4.2 that participants in *any* of the treatments could potentially get credit for process descriptions that had correct process concepts, e.g., using combinations of features.)

4.4.1.1 Completeness and intelligibility types

Two of the intelligibility types, *why* and *input*, relate to features, and participants tended to do better at understanding features than process (Figure 4.8). However, a closer look at *which* participants did better suggests that their understanding of features aligned with completeness. For example, participants in the high-completeness groups (HH and LSHC) averaged 5.5 valid feature codes per participant, versus the other treatments' average of 4.3. The invalid features added more evidence consistent with this, with

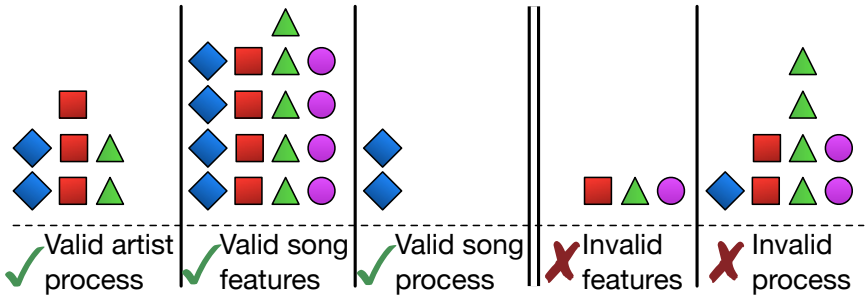


Figure 4.8: Post-task questionnaire results. Each mark is one participant, represented as in Figure 4.7. Only HH participants described all the valid aspects of the recommender (at least one blue diamond in the three left-most bins), and only one made an invalid description (only one blue diamond in the two right-most bins).

high-completeness participants averaging 4.6 invalid features versus other participants' 6.3 invalid features.

Completeness may also have helped participants understand the recommendation process. As Figure 4.9 shows, participants' understanding (as per Table 4.2 codes) of the *artist* recommendation process (explained through the *model* and *why-artist* intelligibility types) tended to increase with the completeness of their treatment. In particular, the *model* explanation was referenced by half of the participants who correctly discussed the artist recommendation process (Figure 4.10). Completeness showed no evidence of impacting participant understanding of the *song* recommendation process; however, this was primarily explained via the Why this Song explanation, and this explanation did not vary in the completeness dimension across treatments.

Recall that we also increased completeness by adding the *certainty* intelligibility type, but this type did not seem to interest participants: only two participants mentioned certainty at all, and each did so only once. Although research has shown that certainty is a useful intelligibility type to users assessing a machine learning system's reliability (Groce et al., 2014), other researchers have found that certainty does not help users' perceived understanding of how a recommender operates (Cramer et al., 2008). Our work suggests that this finding extends to *actual* understanding.

These results suggest that increasing completeness was beneficial to participants' mental models, and that some effective ways to increase completeness included the

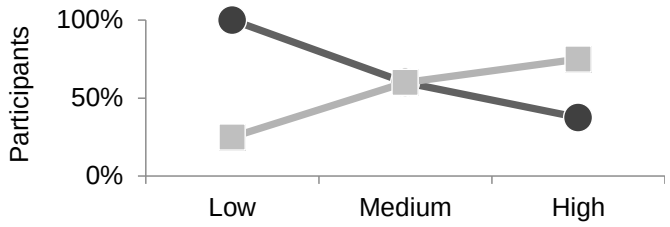


Figure 4.9: Each dot is the percentage of participants who correctly understood the artist recommendation process (as per Table 4.2's codes). More participants understood it as completeness (dark) increased, but fewer participants understood it as soundness (light) increased.

model intelligibility type and the completeness of the *why* type. However, we found no evidence that increasing completeness via *certainty* improved the fidelity of participants' mental models.

4.4.1.2 Soundness and intelligibility types

Although HH participants' performance may at first glance suggest that high soundness was also helpful, looking at soundness in isolation suggests a different story. High-soundness participants (HH and HSLC) showed almost no differences from the other participants in their mentions of valid vs. invalid features or processes. Instead, the clearest pattern was one of *decreased* understanding of the *artist* recommendation process as soundness increased (Figure 4.9).

One hypothesis is that HH and HSLC participants spent most of their attention on their complex Why this Song explanations, causing them to ignore other explanations. Indeed, participants in these high soundness treatments viewed the How it All Works explanation only about half as often as participants in the low-soundness treatment (mean 0.8 vs. 1.4 views per person). Instead, they focused on their complex Why this Song explanations: they viewed these during more songs than participants in the low-soundness treatment (mean of 7.6 vs. 6.3 songs) and often even reviewed prior Why this Song explanations (during an average of 1.9 songs vs. 0.7). One participants in the HH treatment explained why she kept reviewing prior explanations:

> *P9-HH: "The [high-soundness Why this Song] sheet is a little bit hard to look at [flips through prior Why this Song sheets], but I'm just looking for things that I'm seeing, from one song to the next, that are similar, and that it says it's using for matching and making the predictions."*

Further, the high-soundness explanations were associated with over twice as many Information Gap codes, which indicate that as participants viewed these explanations, they had additional questions and expressed more uncertainty as they described why they thought each song had been recommended (mean 7.0 codes per participant) than other treatments (mean 3.3 codes per participant).

These results suggest that increasing soundness was beneficial from a mental model standpoint, but unlike completeness, increasing soundness also carried downsides. The most sound explanations appear to have required more attention from participants—resulting in less attention paid to other explanations—and led to more participant questions and information gaps. However, when soundness and completeness were *both* at our experiment's highest levels, so were participants' mental model scores (Figure 4.7), suggesting that very complete—but unsound—explanations are not enough to help end users understand how machine learning systems operate.

4.4.2 Barriers to developing high-fidelity mental models (RQ4.3)

No participant's understanding of the recommender was perfect: the highest mental model score was 8.4 out of 10 (recall Figure 4.7). We found evidence of two barriers to building high-fidelity mental models; these barriers were shared among all participants, regardless of treatment.

First was participants' incorrect assumptions about the explanations' completeness. Every participant, at some point during their task, incorrectly assumed that the recommender used information that it did not have access to (e.g., the tone of the singer's voice)—even though the input explanation (What the Computer Knows) was complete across all treatments. For example, this high-soundness low-completeness participant had read the What the Computer Knows explanation multiple times before asking:

> *P6-HSLC: "So I guess, does a computer have access to lyrics for a song, does it take that into consideration?" [Facilitator refuses to answer, and participant re-reads*

the What the Computer Knows sheet yet again.] P6-HSLC: "Oh right, so probably not then."

The counts from the post-session questionnaire results were consistent with this phenomenon. In responding to a question asking if the explanations included every important detail about why a song was recommended, the average response was only 13.0 (21 indicating "always", 0 indicating "never"). HH participants, however, responded more positively (mean of 18.0), suggesting that high soundness and high completeness together can help convince users that the explanations *do* discuss everything relevant to the learning system's reasoning.

The second barrier was lack of knowledge of the *process* of how recommendations were made. Participants rarely discussed process, focusing much more heavily on features, as Figure 4.10 illustrates. Some participants even described a single feature as the sole reason for a recommendation:

P2-HH: "Yeah, see, it's all the way at the bottom of the loudness [feature]. So... that's why [it was recommended]."

Features may have been easier for participants to understand because they were explained concretely (i.e., in the context of specific examples). Figure 4.11 shows that participants used the concrete Why this Song and Why this Artist explanations much more than the abstract (i.e., no specific examples) How it All Works and What the Computer Knows explanations.

Note, however, that although our abstract How it All Works explanation was infrequently used, when participants did use it, a larger percentage (50%) correctly discussed the recommendation process than with any other explanation (Figure 4.10). Similarly, participants who used the abstract What the Computer Knows explanation discussed more valid features than invalid features.

Alternatively, participants may have paid the most attention to the Why this Song explanations because it was the only explanation that changed during the experiment. The other explanation types were presented at the beginning of the study and may have attracted less participant attention because they were never updated. Dynamically updating explanations may be one presentation option to draw a user's attention to the full range of explanations in a highly complete system, but this is an open question that requires further investigation.

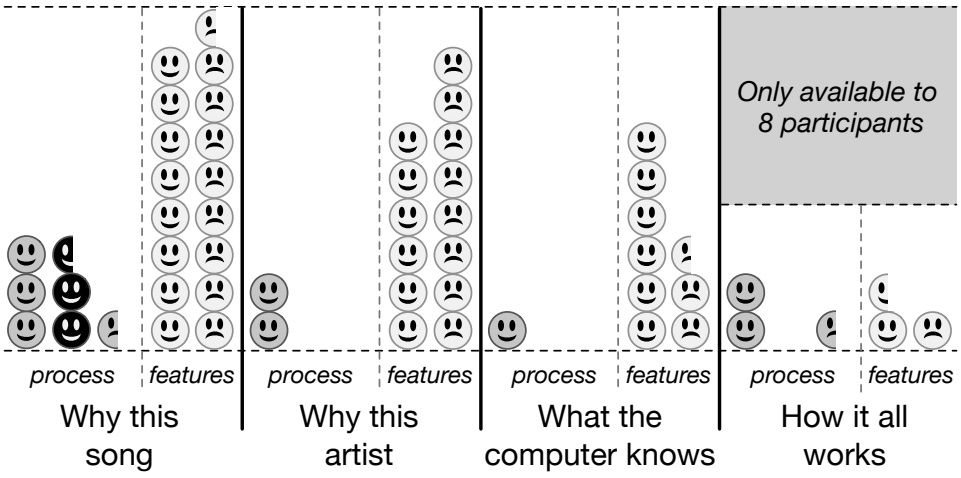


Figure 4.10: Participants giving correct (smiles) and incorrect (frowns) descriptions upon referencing an explanation. Each face represents two participants. (Light): song features. (Medium): artist recommendation process. (Dark): song recommendation process. Both *why* explanations were the most popular, but the What the Computer Knows explanation resulted in the fewest invalid *feature* comments, while the How it All Works explanation had the highest percentage of participants correctly describing the *process*.

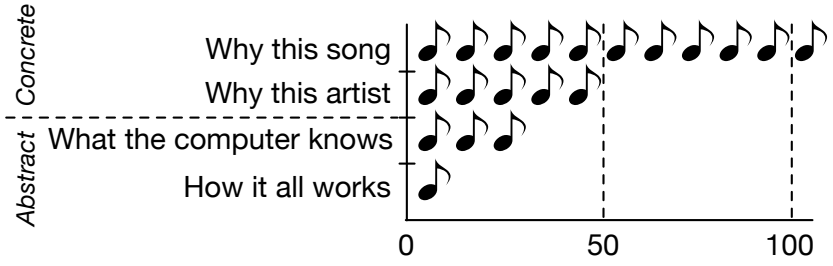


Figure 4.11: Number of times participants referenced each explanation: each music note represents ten references. Participants referenced the Why this Song explanation during almost every recommended song.

4.4.3 Is it worth it? (RQ4.4)

The Attention Investment Model (Blackwell, 2002) predicts that users will use high-cost explanations only if they think the benefits will outweigh the costs. Thus, we investigated participants' perceived benefits (given the perceived costs) using the questions "If the recommendations improve, do you think it is worth the time and effort you spent during this study to give feedback to the recommender?" and "Would you take a similar amount of time as this study to learn similar things about other recommenders you use?" (Each study session lasted less than two hours.) We used the summation of these questions to estimate perceived benefits, and the summation of the NASA-TLX questions about mental demand, effort expended, and frustration/annoyance to estimate costs (each question had a 21-point scale).

As Figure 4.12 shows, the LSHC participants were surprisingly positive about the benefits vs. costs of referring to the explanations—more than three times as positive as participants viewing less complete but more sound explanations (MM and HSLC). We had expected the MM treatment to best balance costs vs. benefits—these participants received explanations that seemed likely to be the easiest to understand at a reasonable cost. However, our results showed that instead, high completeness seemed to be important to our participants. To summarize Figure 4.12, participants in the two high-completeness treatments perceived working with the explanations to be a better cost/benefit proposition than the other treatments' participants did. In contrast, soundness did not seem to be an asset to participants' perception of cost-benefit. This may come back to the lower understanding associated with higher soundness in the absence of high completeness (recall Figure 4.9). One high-soundness low-completeness participant reinforced this point, remarking that the high-soundness explanations *could* have been useful, but she was unable to make much sense of them during the study:

> *P6-HSLC: Probably should have looked at [the Why this Song sheet] more. Facilitator: Do you think this could have been useful? P6-HSLC: Yeah. . . I guess I'm still trying to grasp and understand this whole thing here (points at Why this Song sheet).*

4.4.4 In explanations we trust? (RQ4.5)

To some low-soundness participants, the decision stump in their explanations seemed clearly wrong. For example:

> *P13-LSHC: "It says loudness again, I'm really not understanding why it keeps going back to that and not using energy, or like, anything else."*

To understand participants' perceptions of whether the explanations they viewed were sound and complete, we asked them "Do you think the explanations are accurate about why the recommender chose each song?" (perceived soundness), and "Do you think the explanations are including all of the important information about why the recommender chose each song?" (perceived completeness). We asked about soundness and completeness separately to determine whether participants could discern whether explanations were sound, complete, or both. For example, we hypothesized LSHC participants would rate their explanations as more complete than sound, while HSLC participants would consider their explanations more sound than complete. However, our results suggest participants did not differentiate explanations in this way: the average difference between the two scores was only 1.5 on a 21-point scale, and both LSHC and HSLC participants rated their explanations as slightly more sound than complete.

Because the perceived soundness and completeness scores together form a holistic assessment of trust, we summed them to yield a single trust score. The results, plotted for each participant, are shown in Figure 4.13. The LSHC participants had the three lowest trust ratings, while most HH participants accurately gauged their explanations to be the most sound and most complete. This suggests there is some danger to simplifying explanations by reducing soundness—users may perceive that such explanations do not accurately represent the system's reasoning, and so may distrust (and disregard) them.

4.5 Discussion

Our results suggest that the most sound and most complete explanations (HH) were the most successful at helping participants understand how the learning system worked, and did so with a surprisingly good cost/benefit ratio. Further, HH participants trusted their explanations more than participants in other treatments, particularly LSHC. Indeed, the

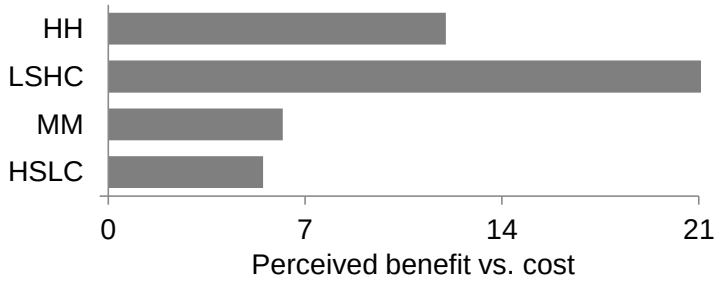


Figure 4.12: Perceived benefit vs. cost scores (*benefit_score* – *cost_score*), averaged by treatment. The high-completeness participants (top two rows) perceived relatively high benefits vs. costs of the explanations.

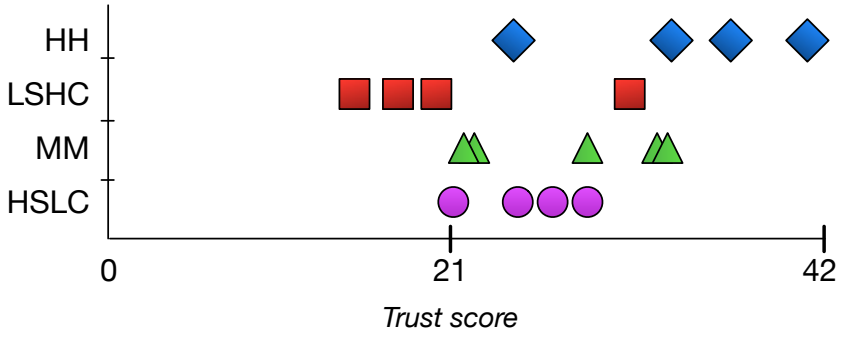


Figure 4.13: Trust scores for each participant. The LSHC treatment's scores were relatively low: these participants accurately rated their explanations as unsound, but also *inaccurately* rated them as incomplete.

main problem we identified with HH was that participants were at risk of focusing on a single complex explanation (e.g., the very detailed Why this Song explanation) to the exclusion of other information.

The story was different when only soundness *or* completeness was at our highest level. High completeness alone (LSHC) provided participants with the best perceived cost/benefit ratio of attending to the explanations, the second-highest average mental model score, and the best understanding of the artist recommendation process. However, these participants placed the least trust in the explanations. High soundness alone (HSLC) did result in more trust, but was also associated with higher perceived costs, lower perceived benefits, and flawed mental models.

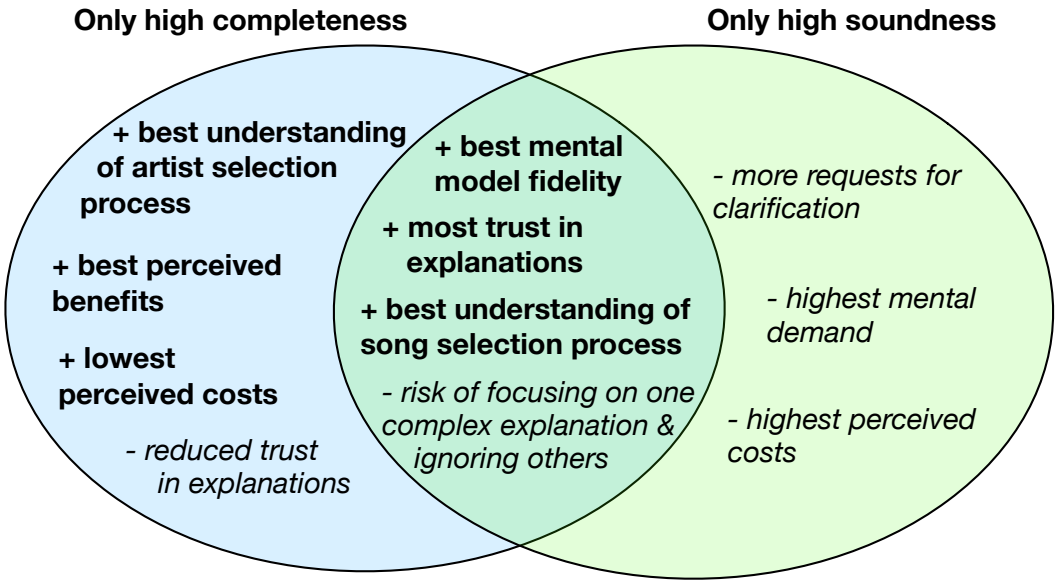


Figure 4.14: The benefits (bold) and costs (italics) of our highest completeness and highest soundness treatments. All of the benefits required high completeness (left and middle), but many of the costs were only observed when soundness was high but completeness was low (right).

Overall, we found that presenting explanations in a sound and complete manner is a surprisingly good design choice, even for relatively low-benefit learning systems such as media and product recommenders. Indeed, we saw a slightly *negative* relationship between mental model fidelity and user satisfaction with the recommendations, suggesting that the hope of improving even such low-benefit system may be sufficient motivation for users to learn more about the system. However, if a designer's user testing of a learning system reveals that its target audience believes such explanations are not worth attending to, our findings suggest that reducing soundness while preserving completeness will improve the cost/benefit ratio of attending to explanations. Figure 4.14 summarizes what tool designers may expect to see when presenting end users (like our participants) with explanations that are very sound, very complete, or both.

4.6 Conclusion

Part of enabling end users to personalize their machine learning systems is explaining these systems to users well enough for them to build high-fidelity mental models. In this chapter we considered two dimensions of explanations—soundness and completeness—and explored how each impacts end users' mental model fidelity, their perceptions of the cost/benefit trade-off of attending to these explanations, and their trust in the explanations. Among our findings were:

RQ4.1 (Soundness and completeness): Our most complete explanations (as in HH and LSHC) were associated with the best mental models; reduced completeness was the shared feature of the two worst-performing treatments (HSLC and MM). Soundness was also important—three of the four participants with the best mental models were part of the high-soundness, high-completeness treatment. The poor performance of the high-soundness, *low*-completeness treatment, however, suggests that explanations should not focus on accurately explaining only a small part of a complex system (soundness); they must also explain how the larger system operates (completeness).

RQ4.2 and RQ4.3 (Explanations and obstacles): Participants had more difficulty understanding the learning system's reasoning process than the features it used, but abstract explanations of the *model* intelligibility type helped overcome this obstacle. However, participants appeared to prefer more concrete explanations (recall Figure 4.11). Finding a way to present intelligibility types, especially the *model* type, in a concrete manner may help to draw user attention to them.

RQ4.4 (Costs and benefits): Our most complete explanations were associated with the highest perceived benefits and lowest perceived costs of learning about the system; completeness even helped moderate the cost of very sound explanations (as in the HH condition).

RQ4.5 (Trust): Participants correctly perceived that the LSHC explanations were unsound, but also refused to trust that these explanations were complete. Participants placed the most trust in HH explanations. Designers who intentionally reduce the soundness or completeness of their system's explanations risk a user base who will not trust—and thus, will not attend to—these explanations.

These findings suggest that many popular machine learning systems offer explanations that are too low in fidelity to enable users to understand how they work, and show how different intelligibility types (e.g., *why*, *model*, *inputs*, etc.) can increase explanation fidelity, and with it the fidelity of users' mental models. Further, our cost/benefit results show that users want to learn more about these systems if their effort is rewarded with the ability to improve the system's predictions or recommendations. Thus, increasing explanation fidelity can be a win/win for end users—motivated users can build a high-fidelity mental model of their machine learning systems, and as Chapter 3 demonstrated, such users can then employ their knowledge to efficiently personalize each system's reasoning. In the following chapters we shall test this hypothesis by designing and evaluating an explanation-centric approach based upon these findings.

Chapter 5: Explanatory Debugging and ElluciDebug

5.1 The principles of Explanatory Debugging

This thesis has shown the potential value of helping users build high-fidelity mental models (Chapter 3) and studied the impact of different types and fidelities of explanation content on mental model development (Chapter 4). We now build upon these findings to propose a new explanation-centric approach—*Explanatory Debugging*[1]—for personalizing interactive machine learning systems.

The crux of Explanatory Debugging is the thesis that *building a better mental model allows for better debugging*. Thus, the first goal of our approach is to help end users build better mental models of a machine learning system. The second goal is to support user feedback that will efficiently adjust the learning system's reasoning. We hypothesize that Explanatory Debugging will help users understand the underlying reason for each of the learning system's mistakes, allowing users to then directly fix that incorrect reasoning.

Explanatory Debugging is defined by two principle features: the ability of the end user to view a machine-generated explanation of why the learning system made each prediction (*explainability*), and the ability of the end user to correct each explanation if he or she disagrees with the learning system's reasoning (*correctability*). These features, in turn, are defined by the following principles:

1. **Explainability: Accurately explain the machine learning system's reasoning.** Our first principle for Explanatory Debugging is *Explainability*: accurately explain the learning system's reasons for each prediction to the end user. This principle builds upon our Chapter 3 finding that users who built better mental models while interacting with the learning system were better able to personalize its reasoning. In a similar vein, other research has found that including explanations of a learning system's predictions helps users to better personalize the system, though these

[1] We say "debugging" because we view personalization of a machine learning system as an end user debugging problem: the user is trying to adjust the computer's reasoning to correct mistakes or support new situations.

works did not explore whether the explanations also helped to improve users' mental models (Kulesza et al., 2010; Kapoor et al., 2010; Bostandjiev et al., 2012). Without explanations, however, research has shown that users struggle to build accurate mental models of such systems (Chapter 3; Tullio et al., 2007; Lim et al., 2009). This suggests *in situ* explanations are a necessary condition to help end users learn how a machine learning system operates. To help users build high-fidelity mental models of the learning system, these explanations should observe the following principles:

1.1 **Be iterative.**
The results of Chapter 3 suggest that end users can best personalize a learning system if they build their mental model *while interacting with it*. Thus, explanations need to support an iterative learning process—designers should not expect a single explanation to suffice (e.g., a video explaining the learning system's model, or a long textual description of how the system works). Instead, iterative explanations should be concise, easily consumable "bites" of information. If a user is interested in learning more about the system, he or she can attend to many of these explanations to iteratively build a higher-fidelity mental model.

Explanations can be made iterative via layering, in which the initial explanation is concise but gives the user the option to view a more detailed explanation on-demand (Storey et al., 1999). Even without layering, however, explanations can fulfill this principle if they allow the user to direct his or her own learning *in situ*. For example, if each of a learning system's predictions is accompanied by a brief explanation, a curious user could examine several explanations in order to identify common themes that may illustrate the system's general decision-making process.

1.2 **Be sound.**
Recall that we define *soundness* as "the extent to which *each component of an explanation's content* is truthful in describing the underlying system", so a sound explanation is not simplified by explaining the model as if it were less complex than it actually is. In Chapter 3 we found a linear correlation

between the improvement of a user's mental model and their ability to control the learning system as desired, suggesting that the more someone learns about the underlying system while interacting with it, the better they will be able to control it. Further, Chapter 4 details the impact of explanation fidelity on mental model development, finding that users did not trust—and thus, were less likely to attend to—the least sound explanations. Because reducing soundness reduces both the potential utility of the explanation and the likelihood that users will invest attention toward it, Explanatory Debugging entails designing explanations that are as sound as practically possible.

One method for evaluating explanation soundness is to compare the explanation with the learning system's mathematical model. How accurately are each of the model's terms explained? If those terms are derived from more complex terms, is the user able to "drill down" to understand those additional terms? The more these explanations reflect the underlying model, the more sound the explanation is.

1.3 **Be complete.**

Recall that we define *completeness* as "the extent to which *all* of the underlying system is described by the explanation", so a complete explanation does not omit important information about the model. In Chapter 4, we found that end users built the best mental models when they had access to the most complete explanations. These explanations informed users of all the information the learning system had at its disposal and how it used that information to make predictions or recommendations. Also pertinent is work showing that users often struggle to understand how different parts of the system interact with each other (Kulesza et al., 2011). Complete explanations that reveal how different parts of the system are interconnected may help users overcome this barrier.

One method for evaluating completeness is via Lim and Dey's *intelligibility types* (Lim and Dey, 2009), with more complete explanations including more of these intelligibility types.

1.4 **But don't overwhelm.**
Balanced against the soundness and completeness principles is the need to remain comprehensible and to engage user attention. Our findings from Chapter 4 suggest that one way to engage user attention is to frame explanations concretely, such as referencing the predicted item and any evidence the learning system employed in its prediction. In some circumstances, selecting a more comprehensible machine learning model may also be appropriate. For example, a neural network can be explained as if it were a decision tree (Craven and Shavlik, 1997), but this reduces soundness because a different model is explained. Similarly, a model with 10,000 features can be explained as if it only used the 10 most discriminative features for each prediction, but this reduces completeness by omitting information that the model uses. Alternative approaches that embody the Explanatory Debugging principles include selecting a machine learning model that can be explained with little abstraction (e.g., Szafron et al., 2003; Lacave and Díez, 2002; Stumpf et al., 2009) or using feature selection techniques (Yang and Pedersen, 1997) in high-dimensionality domains to prevent users from struggling to identify which features to adjust (as happened in (Kulesza et al., 2011)).

2. **Correctability: The explanation *is* the feedback mechanism.**
Our second top-level principle for Explanatory Debugging is *Correctability*: allow users to explain corrections back to the learning system. To enable an iterative cycle of explanations between the system and the user, in Explanatory Debugging the machine-to-user explanation should also serve as the user-to-machine explanation. Research suggests that to elicit corrections from users, this feedback mechanism should embody the following principles:

2.1 **Be actionable.**
Both theory (Blackwell, 2002) and prior empirical findings by ourselves (Chapter 3) and others (Bunt et al., 2012) suggest end users will ignore explanations when the benefits of attending to them are unclear. By making the explanation actionable, we hope to lower the perceived cost of attending to it by obviating the need to transfer knowledge from one part of the user interface (the explanation) to another (the feedback mechanism). Actionable

explanations also fulfill three aspects of Minimalist Instruction (van der Meij and Carroll, 1998): (1) people are learning while they perform real work; (2) the explanatory material is tightly coupled to the system's current state; and (3) people can leverage their existing knowledge by adjusting the explanation to match their own mental reasoning. As Minimalist Instruction has been successfully applied to complex problem domains (e.g., learning a programming language (Rosson et al., 1990)), we hypothesize that actionable explanations embodying aspects of Minimalist Instruction will likewise help users problem solve in the domain of machine learning.

2.2 **Be reversible.**
A risk in enabling users to provide feedback to a machine learning system is that they may actually make its predictions worse (e.g., Stumpf et al., 2009; Kulesza et al., 2010). Being able to easily reverse a harmful action can help mitigate this risk. It may also encourage self-directed tinkering, which can facilitate learning (Rowe, 1973). When combined with Principle 2.1 (Be actionable), reversibility also fulfills a fourth aspect of Minimalist Instruction (van der Meij and Carroll, 1998): help people identify and recover from errors.

2.3 **Always honor user feedback.**
As Yang and Newman found when studying users of learning thermostats (Yang and Newman, 2013), a system that appears to disregard user feedback deters users from continuing to provide feedback. However, methods for honoring user feedback are not always straightforward. Handling user feedback over time (e.g., what if new instance-based feedback contradicts old instance-based feedback?) and balancing different types of feedback (e.g., instance-based feedback versus feature-based feedback) requires careful consideration of how the user's feedback will be integrated into the learning system.

2.4 **Incremental changes matter.**
In Chapter 4, participants claimed they would attend to explanations *only if* doing so would enable them to more successfully control the learning system's

predictions. Thus, continued user interaction may depend upon users being able to see incremental changes to the learning system's reasoning after each interaction (i.e., overcoming the *gulf of evaluation* that exists between a user's mental model and a system's actual state (Norman, 2002)). This principle is also related to Principle 1.1 (Be iterative) because our thesis is that users will develop better mental models iteratively, requiring many interactions with the learning system. These interactions may not always result in large, obvious changes, so being able to communicate the small, incremental changes a user's feedback has upon a learning system may be critical to our approach's feasibility.

We next present a prototype that embodies the above principles in the domain of text classification.

5.2 EluciDebug: A prototype instantiating Explanatory Debugging

We instantiated the Explanatory Debugging principles in EluciDebug, a text classification prototype we developed to evaluate the approach's effectiveness. We chose text classification because (1) many real-world systems require it (e.g., spam filtering, news recommendation, serving relevant ads, search result ranking, etc.) and (2) it can be evaluated with documents about common topics (e.g., popular sports), allowing a large population of participants for our evaluation. We designed EluciDebug to look like an email program with multiple folders, each representing a particular topic. The prototype's machine learning component attempts to automatically classify new messages into the appropriate folder.

By embodying the Explanatory Debugging principles set forth in Section 5.1 we ensured that the design of EluciDebug was grounded in theory, but we also needed to ensure that its design worked well in practice. Thus we employed a participatory design methodology (Shneiderman and Plaisant, 2010) to test our ideas for instantiating Explanatory Debugging. For each design iteration we ensured that the prototype embodied the principles of Explanatory Debugging (Section 5.1), then we asked an end user to accomplish several tasks with the prototype. The initial iterations involved paper prototypes to encourage rich participant feedback; Figure 5.1 shows one such paper

prototype from early in the design process. A researcher observed participants and noted each of their suggestions and any misunderstandings or problems they encountered, then incorporated possible solutions into the next iteration. The paper prototypes successfully elicited major suggestions for refinement, such as allowing the user to see a message and the explanation for its prediction simultaneously, consolidating the three tabs of the *Important words* explanation into a single view, and adding indictors to highlight incremental changes. Participants' suggestions and misunderstandings grew steadily less with each iteration, so after four iterations we implemented a high-fidelity prototype in C# and the .NET Framework.

We conducted additional usability tests with our high-fidelity prototype, though as expected, the feedback about our high-fidelity prototype focused on progressively more minor behavioral adjustments. For each test, an end user was given a short tutorial on the ElluciDebug user interface, then asked to make its predictions as accurate as possible. A researcher observed each test and took notes about how participants used the prototype. We revised the prototype after each test to resolve participants' problems. The final prototype—following four iterations—is shown in Figure 5.2.

While the principles of Explanatory Debugging primarily deal with the user interface, two principles place constraints on the machine learning model: (1) it must be able to *honor user feedback* in real-time, and (2) it must be explainable with enough soundness and completeness to allow users to build high-fidelity mental models of how it operates *without overwhelming* them. Our ElluciDebug prototype uses a multinomial naive Bayes model (MNB) (Kibriya et al., 2004) with feature selection (Yang and Pedersen, 1997) to meet these constraints. Evaluating the suitability of—or changes necessary to—using Explanatory Debugging with other machine learning models remains an open question.

5.2.1 The multinomial naive Bayes classifier: A brief review

Before describing how we integrated MNB with Explanatory Debugging, we first summarize how MNB operates. An MNB classifier computes the probability that a given input (e.g., the document being classified) has of belonging to each output (e.g., the possible labels). The output with the highest probability "wins" and becomes the predicted label for the input. For example, if an MNB classifier calculates that a document has a 70% probability of being junk mail and a 30% probability of not being junk mail, the

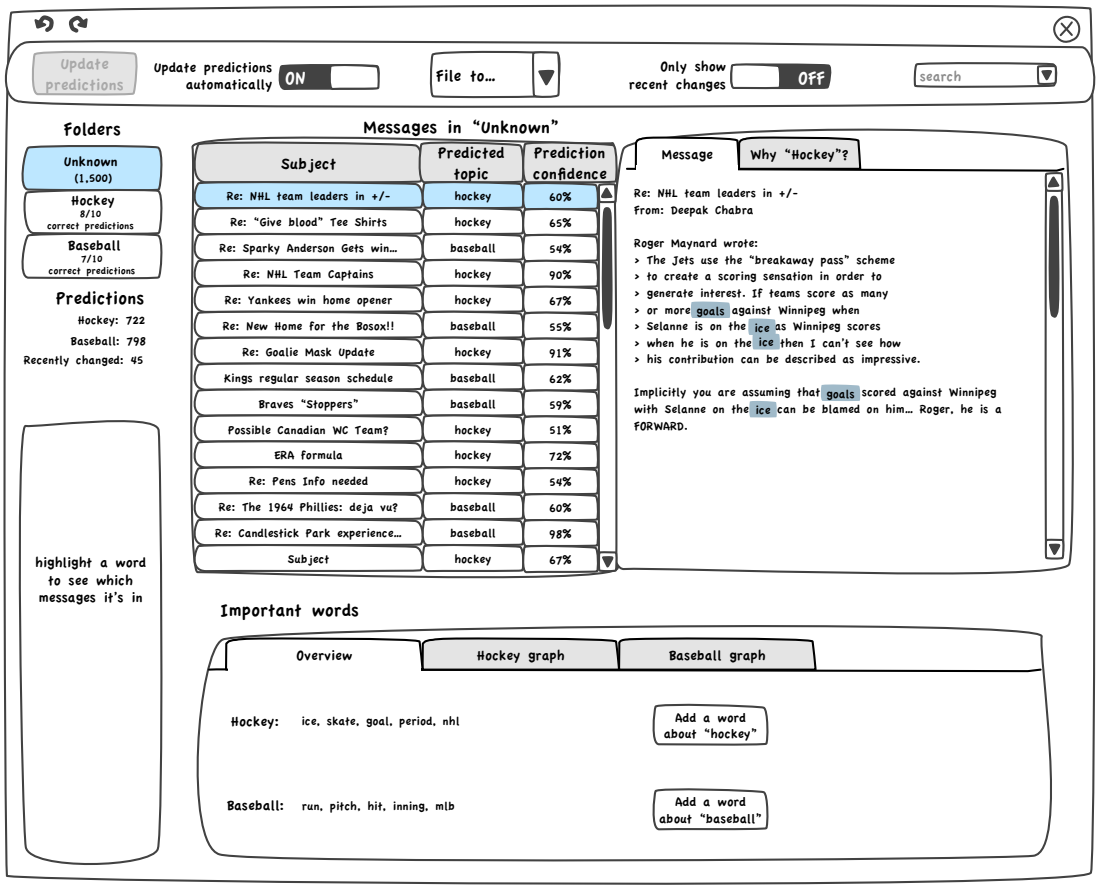


Figure 5.1: An example of the paper prototype that participants used to provide feedback early in the design process. Over a dozen different variations were prepared for each session, allowing the researcher to show each participant how the software would respond to their different actions.

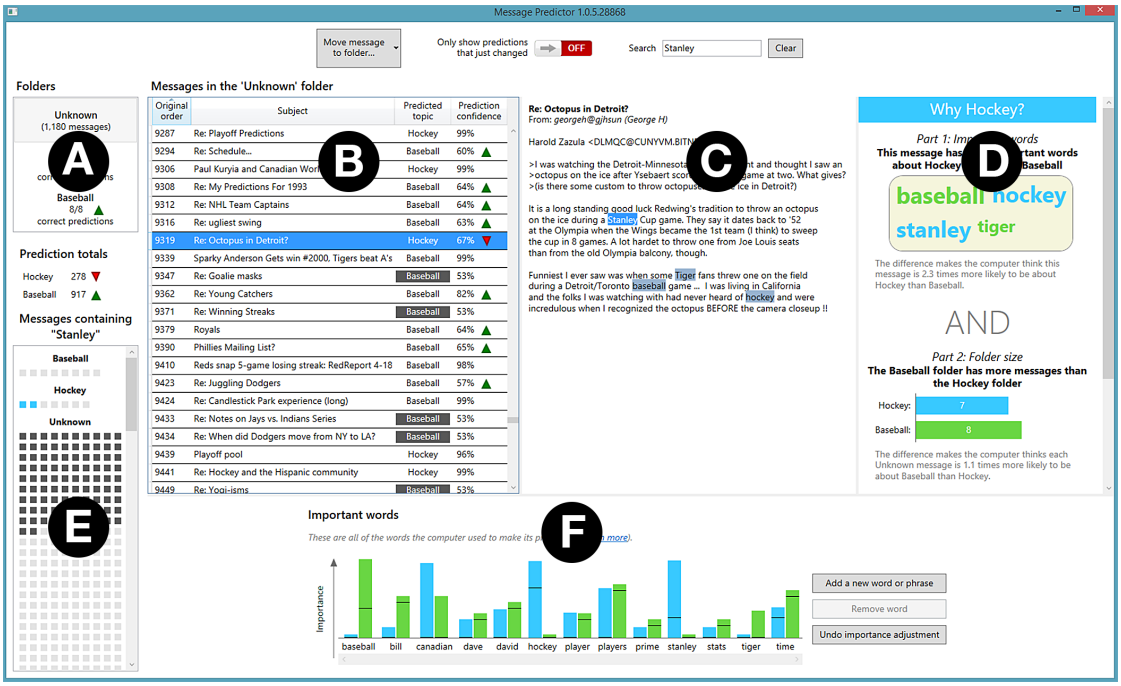


Figure 5.2: The final version of ELUCIDEBUG following eight rounds of revisions (four with low-fidelity prototypes and four with high-fidelity prototypes) to the initial paper prototype shown in Figure 5.1. (A) List of folders. (B) List of messages in the selected folder. (C) The selected message. (D) Explanation of the selected message's predicted folder. (E) Overview of which messages contain the selected word. (F) Complete list of words the learning system uses to make predictions.

document will be labeled as junk mail. The equations for computing probability, as defined in (Kibriya et al., 2004), are shown below. We use c to represent an individual class in the collection of potential output classes C, d_i to represent an individual document to classify, and assume that the only features the classifier uses are individual words in the set of known documents:

$$\Pr(c|d_i) = \frac{\Pr(c)\Pr(d_i|c)}{\Pr(d_i)} \tag{5.1}$$

The term $\Pr(c)$ represents the probability that any given document belongs to class c and can be estimated by dividing the number of documents in c by the total number of

documents in the training set. The term $\Pr(d_i|c)$ represents the probability of document d_i given class c and can be estimated as:

$$\Pr(d_i|c) = \prod_n \Pr(w_n|c)^{f_{ni}} \tag{5.2}$$

The term f_{ni} is the number of instances of word n in document d_i and the term $\Pr(w_n|c)$ is the probability of word n given class c, estimated with the equation

$$\Pr(w_n|c) = \frac{p_{nc} + F_{nc}}{\sum_{x=1}^{N} p_{xc} + \sum_{x=1}^{N} F_{xc}} \tag{5.3}$$

where F_{nc} is the number of instances of word n in all of the training documents for class c, N is the number of unique words in the training documents for all classes, and p_{nc} is a smoothing term (usually 1) to prevent the equation from yielding 0 if no documents from class c contain word w_n.

5.2.2 The Explanatory Debugging principles in ElluciDebug

5.2.2.1 Being iterative

To help end users iteratively build better mental models, ElluciDebug employs two strategies: (1) each explanation focuses on an individual component or aspect of the learning system, and (2) layered explanations are available if the user wants more details about certain components. For example, the *Why* explanation (discussed in Section 5.2.2.2) primarily tells users about the learning system's reasons for making a *specific* prediction. Thus, each time the user is curious about why the system made a given prediction, he or she can view this explanation. Over time, the user may learn more about the system's general operation by observing commonalities in the reasons for multiple predictions. Further, this *Why* explanation is layered. Part of the explanation shows users which words contributed to the learning system's prediction—if the user wants to know more about why the system associated a particular word with the given topic, hovering over the word yields a tooltip that explains how the system determines which topic a word is associated with.

5.2.2.2 Being sound

Soundness means that everything an explanation says is true. Our explanations aim to be sound by accurately disclosing all of the features the classifier used to make its prediction, as well as how each feature contributed to the prediction. ElsuciDebug's *Why* explanation is responsible for communicating much of this information to users (Figure 5.3).

Soundly explaining the MNB classifier requires explaining the $\Pr(c)$ and $\Pr(d_i|c)$ terms from Equation 5.1, as these are the only terms that impact the model's predictions.[2] Because the $\Pr(d_i|c)$ term expands into Equation 5.2, we also need to explain how word probabilities for each class factor into the prediction. We can further increase soundness by explaining how these probabilities are calculated (i.e., by explaining Equation 5.3).

In ElsuciDebug we explain the $\Pr(d_i|c)$ term by using a word cloud to visualize the difference between each feature's probability for the two output classifications (Equation 5.3) as the ratio

$$\frac{\Pr(w_n|c_1)^{f_{ni}}}{\Pr(w_n|c_2)^{f_{ni}}} \tag{5.4}$$

where c_1 is the class with the larger probability for feature w_n. We use the result to compute the feature's font size, while its font color is based on the class with the larger probability. For example, in Figure 5.3 the word *stanley* is larger than *tiger* because its ratio of word probability is correspondingly larger, and it is blue because its probability of occurring in the *hockey* class is larger than its probability of occurring in the *baseball* class. Hovering over a word in this cloud shows a tooltip that explains the word's size was determined by a combination of (1) how many times it appeared in each class, and (2) any adjustments the user made to the word's importance (Equation 5.3).

The second component of a sound explanation of the MNB classifier is the $\Pr(c)$ term from Equation 5.1. We explain this term via a bar graph visualization of the number of items in each class (Figure 5.3, middle).

The top-to-bottom design of this entire *Why* explanation, along with the text that describes *Part 1* and *Part 2*, is intended to teach the user that both word presence (the $\Pr(d_i|c)$ term) and folder size (the $\Pr(c)$ term) play a role in each of the classifier's

[2] The $\Pr(d_i)$ term in the denominator of Equation 5.1 is only used to normalize the result to fall within the range 0–1; it does not impact the model's prediction.

predictions. The result, shown at the bottom of Figure 5.3, explains how these two parts are combined to determine the classifier's certainty in its prediction.

5.2.2.3 Being complete

Completeness means telling the *whole* truth. For the MNB classifier, this means not only explaining each of the terms from Equation 5.1, but also all of the information the classifier could potentially use (i.e., the entire feature set the classifier supports), where this information comes from (e.g., does a movie recommender include user-submitted tags of a movie's genre, or only the studio's genre description?), and how likely it is that each prediction is correct. To help ensure completeness, we turn to Lim and Dey's schema of intelligibility types. The results of Chapter 4 suggest that a complete explanation should include Lim and Dey's *why*, *inputs*, and *model* types. Our prior work on end-user testing of machine learning systems suggests that Lim and Dey's *certainty* intelligibility type is critical for assessing the reliability of a machine learning system (Groce et al., 2014), so we also included this type. Finally, Lim's work suggests the usefulness of the *what if* type in scenarios where the user is attempting to change the behavior of a classifier (Lim, 2012), so we have included this intelligibility type as well.

We thus designed our EluciDebug explanations to detail all of the information the classifier *might* use when making predictions, even if that information wasn't relevant for the currently selected prediction. The *Why* explanation shown in Figure 5.3 tells users that both feature presence and folder size played a role in each prediction (the numerator of Equation 5.1). The *Important words* explanations (Figure 5.6) goes even further, telling the user *all* of the features the classifier knows about and may use in its predictions. Because it tells the user about the sources of information available to the classifier, this is also an instantiation of Lim and Dey's *inputs* intelligibility type. To make it clear to users that these features can occur in all parts of the document—message body, subject line, and sender—EluciDebug highlights features in the context of each message (Figure 5.2, part C).

In Chapter 4 we found that Lim and Dey's *model* intelligibility type was associated with better mental models, but this intelligibility type was rarely attended to by most participants. To solve this dilemma, in EluciDebug we incorporated the *model* content into our *Why* explanation—it explains all of the evidence the classifier used, but it also

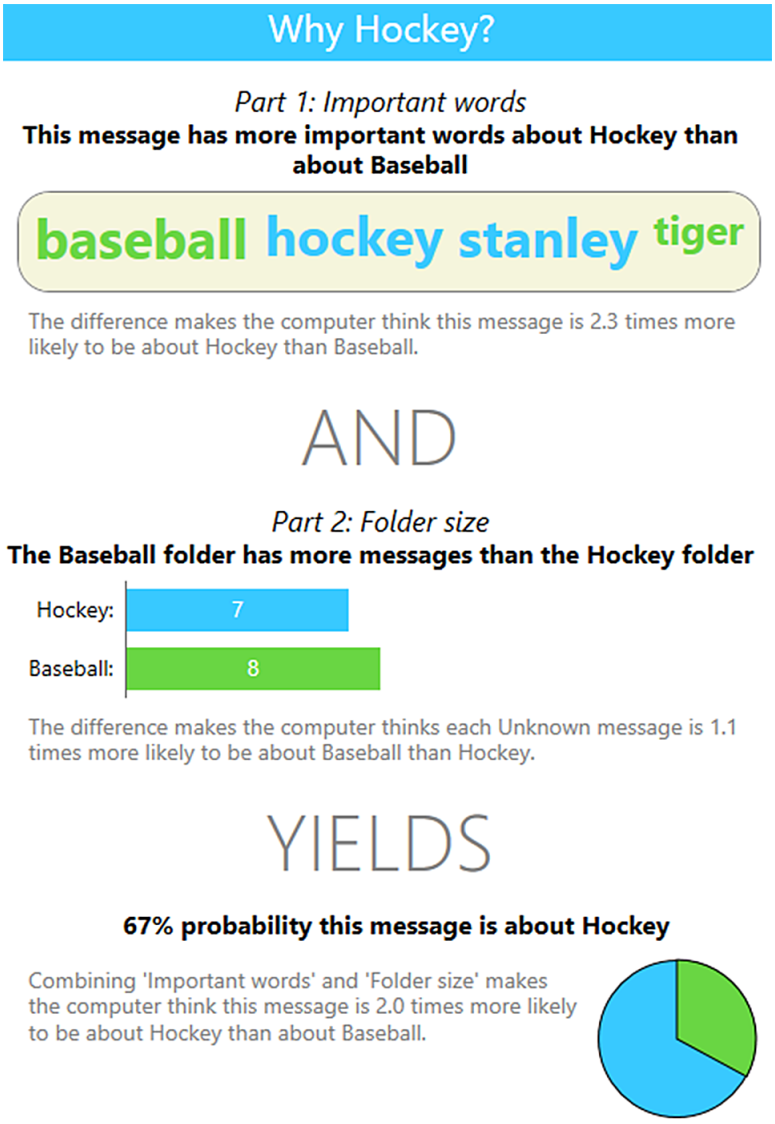


Figure 5.3: The *Why* explanation tells users how features and folder size were used to predict each message's topic. This figure is a close-up of Figure 5.2 part D.

explains where that evidence came from (e.g., words or folder size) and how it was combined to arrive at a final prediction. This approach has the added advantage of making the potentially abstract *model* intelligibility type very concrete; it is now tailored to each specific prediction.

A further aspect of completeness is evident in the *Feature overview* explanation (Figure 5.4). This explanation shows users how many messages contain a given feature and is intended to help users identify when a feature they think the computer should pay attention to may not be very useful. The explanation updates in real-time as the user types in a potential feature; users do not need to add the feature to view its potential impact on classifications, making this an instance of the *what if* intelligibility type. For example, one pilot participant remarked that he thought "AL" would be a useful feature for identifying American League baseball messages, but realized from this explanation that (1) some hockey players are named "Al", and (2) the classifier wasn't case-sensitive, causing it to consider the name "Al" as equivalent to the American League abbreviation "AL", and so decided against adding "AL" to the list of important words.

Finally, we also included the *certainty* intelligibility type. This is instantiated via the *Prediction confidence* column (Figure 5.2, part B), which reveals the classifier's confidence in each of its predictions to the user.

5.2.2.4 Not overwhelming

To avoid overwhelming users, EluciDebug limits the initial set of features available to the classifier using information gain (Yang and Pedersen, 1997). Because Principle 1.3 states that explanations should be as complete as possible, users should be able to see all of the classifier's features. Given this constraint, we decided 50 would be the upper limit on feature set size. Offline tests, however, revealed that the F_1 score of an MNB classifier operating on the 20 Newsgroup dataset did not improve while the feature set size increased from 10 to 50 (Figure 5.5), so we decided our classifier would automatically select only the 10 features with the highest information gain (until the user specifies otherwise by adding or removing features).

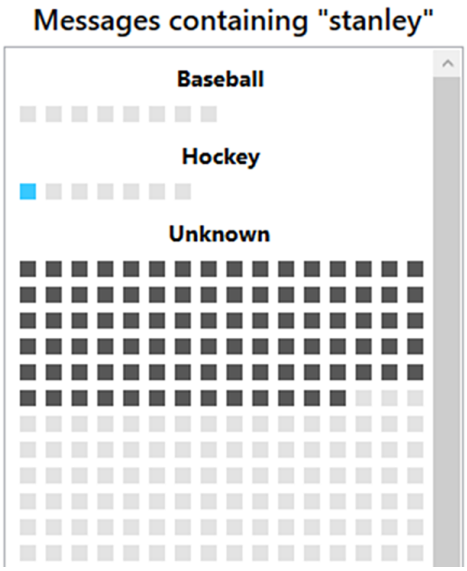


Figure 5.4: The *Feature overview* explanation shows users how prevalent each feature is in the dataset. Each shaded mark represents one message, and users can click on a mark to view the message. This figure is a close-up of Figure 5.2 part E.

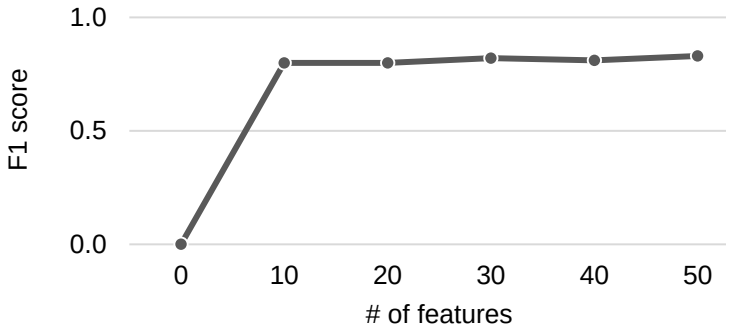


Figure 5.5: Selecting the 10 highest information gain features resulted in similar classifier performance as larger feature sets.

5.2.2.5 Being actionable

The *Important words* explanation (Figure 5.6) is the most actionable of ElluciDebug's explanations. Users can add words to—and remove words from—this explanation, which in turn will add those words to (or remove them from) the machine learning model's feature set. Users are also able to adjust the importance of each word in the explanation by dragging the word's bar higher (to make it more important) or lower (to make it less important), which then alters the corresponding feature's weight in the learning model (this processes will be explained in more detail in Section 5.2.2.7).

The *Why* explanation (Figure 5.3) is another likely candidate for actionability, but we have not yet made it actionable in ElluciDebug. As this explanation includes only features extant in the selected message, it cannot replace the *Important words* explanation because doing so would interfere with our explanations' completeness. It could, however, complement the *Important words* explanation by allowing users to directly adjust the importance of features responsible for the given prediction. For example, users could drag out from the center of a word to increase its importance, or drag in toward the center of the word to decrease its importance. Whether such additional actionability would help users, however, remains an open question.

5.2.2.6 Being reversible

ElluciDebug includes an *undo* button for reversing changes to the *Important words* explanation. There is no limit on how many actions can be un-done because we want users to interact with the system without fear they will harm its predictions; regardless of how much they adjust its reasoning, they can always return to any prior state.

5.2.2.7 Honoring user feedback

ElluciDebug allows users to provide two types of feedback: traditional *instanced-based feedback*, where the user applies a label[3] to an entire item, and *feature-based feedback*, where the user tells the classifier an item should be labeled in a certain manner because of specific features it contains or feature values it matches. ElluciDebug honors instance-

[3]Recall that a *label* is a potential output of the learning system, such as *junk mail* or *normal mail*.

based feedback in a straightforward manner: once the user labels an item, the classifier will use it as part of its training set, with no distinction between older versus more recent feedback. Honoring feature-based feedback, however, is more complicated.

The smoothing term p_{nc} from Equation 5.3 acts as a Bayesian prior, effectively adding some number of virtual occurrences (traditionally 1) to the number of actual occurrences of each word in the training data, and we can leverage it to integrate feature-based feedback. By allowing the user to set the value of p_{nc}, we are letting the user increase the number of virtual occurrences of word n in class c. The result is a classifier that considers word n to be stronger evidence in favor of class c than it had before the user's feedback.

Using the smoothing term as a feature-based feedback mechanism, however, has a drawback: F_{nc} may increase as the training set size increases, causing the value of p_{nc} to become a smaller component of Equation 5.3. Thus, a user's feature-based feedback could account for less and less of the classifier's reasoning as their instance-based feedback increased.

To prevent a user's feature-based feedback from degrading in importance over time, we developed a visual feedback mechanism (Figure 5.6) that allows users to specify how important their feedback should be relative to the model's internal word probabilities (the F terms in Equation 5.3). The black lines on the blue and green bars in Figure 5.6 show the model-computed probabilities for each feature, which serve as a starting point for feature-based feedback. Users can tell the system that the probability of seeing word w_n in class c should be increased by clicking and dragging its bar higher, which will translate to an increased value for p_{nc}. If the user later provides additional instance-based feedback (thus causing F_{nc} to change), p_{nc} will be automatically recalculated such that the ratios of $\sum_{x=1}^{N} p_{xc}$ to $\sum_{x=1}^{N} F_{xc}$ and p_{nc} to F_{nc} remain constant.

5.2.2.8 Revealing incremental changes

There are many components of EluciDebug that may change after each user action, so to avoid confusing users with several different explanation paradigms, we designed a method that consistently reveals changes in terms of *increases* and *decreases*. Increases of any numeric value are identified by green "up" arrows, while decreasing numeric values are identified by red "down" arrows. Examples of each are shown in Figure 5.7.

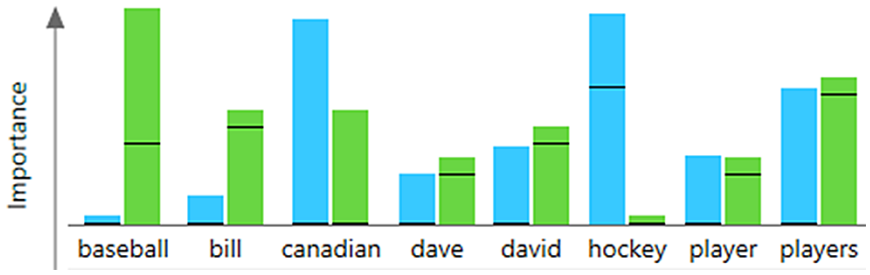


Figure 5.6: The *Important words* explanation tells users all of the features the classifier is aware of, and also lets users add, remove, and adjust these features. Each topic is color-coded (here, blue for *hockey* and green for *baseball*) with the difference in bar heights reflecting the difference in the word's probability with respect to each topic (e.g., the word *canadian* is roughly twice as likely to appear in a document about *hockey* as one about *baseball*, while the word *player* is about equally likely to appear in either topic). This figure is an excerpt from Figure 5.2 part F.

Hovering over either of these arrow icons yields a tooltip detailing what just changed and how much it changed by, e.g., "Confidence increased by 9%". These indicators reveal changes in the number of messages correctly classified in each folder, the total number of messages the machine learning model currently classified into each folder, and the confidence of each prediction.

In addition to numeric change indicators, we also needed an ordinal change indicator to highlight when a message's prediction flipped from one topic to the other. We used a grey background for these recently-changed predictions (Figure 5.7) and included a tooltip explaining that the user's last action resulted in the message's predicted topic changing.

5.3 Conclusion

The instantiation of the Explanatory Debugging principles presented above is one example how Explanatory Debugging can be implemented, but we do not intend to imply that all instances of Explanatory Debugging should look like EluciDebug. For example, a machine learning system that uses a decision tree model may need to replace the *Important words* explanation with a control flow diagram or series of *if/then/else* statements. To hold true to Explanatory Debugging principles, however, the revised

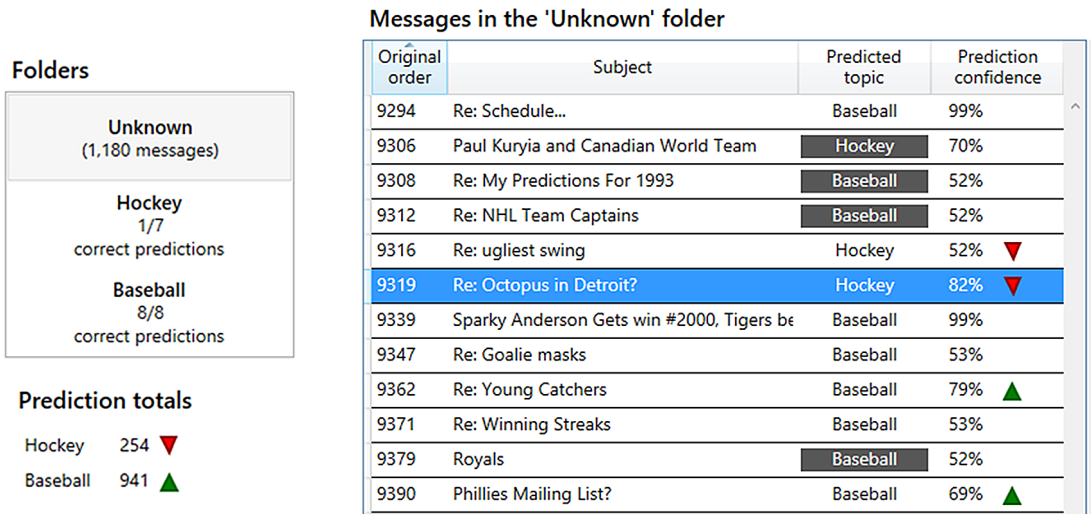


Figure 5.7: ElucidDebug includes several devices to help users overcome the gulf of evaluation. The *Folders* explanation (left, top) shows users how many messages are correctly classified in each folder, while the *Prediction totals* explanation (left, bottom) shows how many messages in the "Unknown" folder are predicted as being about each topic. (Right) The *Prediction confidence* explanation shows users how certain the machine learning system is in each prediction. All three explanations tell the user whether the number has recently increased (a green up arrow) or decreased (a red down arrow), and a tooltip on the arrow tells users the exact magnitude of the change.

explanation needs to accurately reflect the underlying machine learning model and be correctible by end users. What such explanations may look like for different learning models remains an open question.

Chapter 6: Evaluation

To investigate Explanatory Debugging's effectiveness with end users, we evaluated our approach—as instantiated in EluciDebug—via the following research questions:

RQ6.1: Does Explanatory Debugging help end users personalize a classifier *more efficiently* than instance labeling?

RQ6.2: Does Explanatory Debugging help end users personalize a classifier *more accurately* than instance labeling?

RQ6.3: Does Explanatory Debugging help end users build *better mental models* than a traditional black-box machine learning approach?

6.1 Methodology

6.1.1 Experiment design

We used a between-subject, single-factor experimental design to evaluate Explanatory Debugging. The factor we varied was experiment condition: one condition (*control*) used a variation of EluciDebug with all of its explanation and feature-based feedback capabilities removed (Figure 6.1), while the second condition (*treatment*) used the EluciDebug prototype described in Chapter 5. In both conditions EluciDebug was set up as a binary classifier that attempted to predict whether each newsgroup message belonged to one of two topics.

To study the accuracy of a machine learning classifier, we need to know what its predictions *should* be—what is commonly called the *gold standard* or *ground truth*. Thus we can compare each prediction that the classifier makes to what the gold standard says the prediction should be. We also needed a dataset with a sufficient number of items in each category. To make manually sorting every item impractical, we decided that our dataset would need a minimum of 500 items per category.

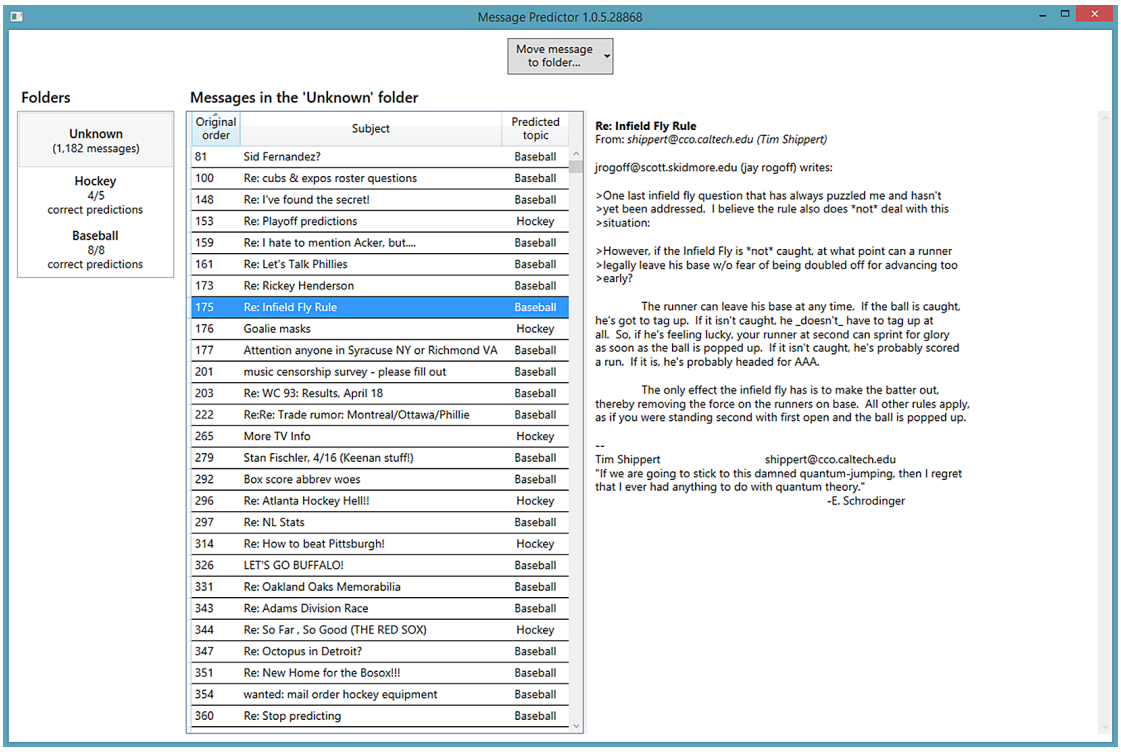


Figure 6.1: Control participants used this variant of EluciDebug, which lacks explanations and feature-based feedback.

We selected the 20 Newsgroups dataset[1] because it provides a gold standard, includes concepts with a low degree of subjectivity (e.g., hockey, baseball, motorcycles, medicine, etc.), and includes 18,846 items approximately evenly distributed across its 20 concepts. This dataset has also been used extensively in machine learning research, giving us many examples of how different machine learning models perform on it.

For this experiment we selected four concepts from 20 Newsgroups: *medicine*, *outer space*, *hockey*, and *baseball* (the sci.med, sci.space, rec.sport.hockey, and rec.sport.baseball newgroups, respectively). We used *medicine* and *outer space* as the topics for the tutorial, and *hockey* and *baseball* as the topics for the experiment task. Each task used two subgroups of a larger group of related concepts (e.g., the *medicine* and *outer space* subgroups are both part of the *science* group) to ensure that there would be some overlap

[1] `http://qwone.com/~jason/20Newsgroups/`

in the terminology of each concept, such as how "player" and "team" may be equally representative of *baseball* or *hockey*. These shared terms help to make the classification task more challenging.

The 20 Newgroups dataset is pre-split into two parts: *training* and *testing*. We used the *training* part to populate our prototype with messages and reserved the *testing* portion as a validation set, meaning that while participants never saw it, we used it in later, offline analyses to evaluate classifier accuracy. The purpose of a validation set is to ensure that a classifier has not overfit itself to the data it has seen, which would become evident by the classifier performing worse on the unseen validation set than on the data it had seen.

To simulate a situation where personalization would be required because sufficient training data does not yet exist, we severely limited the size of the machine learning training set for this experiment. At the start of the experiment this training set consisted of 5 messages in the *Hockey* folder and 5 messages in the *Baseball* folder, with 1,185 unlabeled messages in the *Unknown* folder. The small training set (10 messages) allowed us to evaluate a situation with limited training data, and the large amount of *potential* training data (1,185 messages) allowed us to contrast Explanatory Debugging against black-box instance labeling in a situation where instance labeling could be expected to eventually succeed.

6.1.2 Participants and procedure

We recruited 77 participants from the local community and university. To ensure that participants would have little or no prior experience with software debugging or machine learning, we did not accept participants who had more programming experience than an introductory-level college course. The average age of our participants was 23 years and included 27 females and 50 males.

The experiment session was conducted in groups of up to 10 participants at a time. We assigned each participant to the first available experiment session that met their time constraints and then randomly assigned each session to either the control or treatment condition. A total of 37 participants experienced the control condition and 40 participants took part in the treatment condition.

To ensure that our control and treatment samples were similar to one another, we collected demographic data from participants upon their arrival for the experiment session. This included information such as age, college GPA, and familiarity with the topics for the experimental task, as well as participants' computer self-efficacy, which we assessed via the validated questionnaire in (Compeau and Higgins, 1995). We later analyzed this data using Wilcoxon signed rank tests and found no evidence of differences between our two participant groups.

A researcher introduced participants to the prototype via a brief hands-on tutorial that explained how to use it, but did not discuss how it made predictions. Participants then had three minutes to explore the prototype on their own. To avoid learning effects about the topics, the tutorial and practice session involved messages about different topics (*outer space* and *medicine*) than the main experiment task.

The main experiment task followed the practice session. Participants were asked to "make the computer's predictions as accurate as possible" and given 30 minutes to work. The software logged all participant interactions and logged evaluations of its internal classifier at 30-second intervals.

After the main task concluded, we assessed participants' mental models via a questionnaire. This test instrument evaluated how well participants understood the two components that contribute to the MNB classifier's predictions: feature presence and class ratios. Because feature presence can be easy to detect given certain words (e.g., the word "hockey" is obviously related to the concept of *hockey*), we evaluated participants' understanding of feature presence using both "obvious" and "subtle" features. We define "subtle" features as words that are not normally associated with a topic, but appear in the classifier's training set and thus will impact classification. Participants were given three short messages about two topics (*swimming* and *tennis*) and told that "these are the only messages the software has learned from". Participants in the treatment condition were also given an *Important words* explanation similar to the one they saw in the prototype (Figure 6.2). A second sheet displayed 12 messages, each only one or two sentences long, and asked participants which topic the classifier would assign to each message, and why. To reduce the chance that participants' mental models would improve as a result of extensively reflecting upon the contents of the test instrument, participants were given only 10 minutes to complete the mental model portion of the questionnaire. The

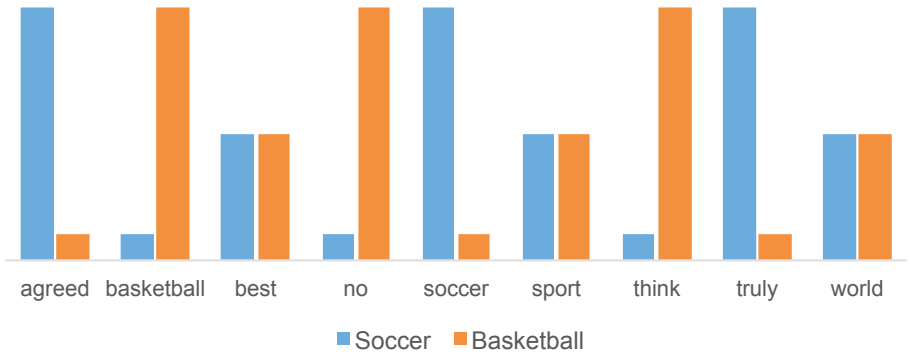


Figure 6.2: Participants in the treatment condition saw this explanation during their mental model assessment, which is almost identical to the explanations they saw while working with the prototype (shown in Figure 5.6).

messages were constructed such that only one component—either an obvious feature, a subtle feature, or class ratios—was entirely responsible for the classification.

To understand participants' reactions to ElucidDebug, the post-task questionnaire also asked participants about various features of the prototype and their perceived task load during the experiment (via the validated NASA-TLX questionnaire (Hart and Staveland, 1988)). All of the materials used in this study have been reproduced in Appendix A.

6.1.3 Data analysis

We used non-parametric methods for all statistical analyses. As suggested in (McCrum-Gardner, 2008), we used Mann–Whitney U-tests for interval and ordinal data, and Spearman's ρ for correlations.

To analyze participants' mental models, a researcher graded participant responses to the post-task mental model questionnaires. Because some participants may have randomly guessed which topic the classifier would predict for each message, we ignored all predicted topics and only graded the *reason* participants stated for the classifier's prediction. Participants earned two points for correct reasons and one point for partially correct reasons. The researcher graded participant responses without knowing which condition the participant was in (i.e., blindly). Each participant's points were summed to

Model component	Answer mentions...	Points
Keywords	... only correct keyword	2
	... only correct keyword and correct class imbalance	2
	... correct keyword plus something else	1
	... a keyword matches, but doesn't specify	1
	... word similarity, but doesn't specify	1
	... correct class imbalance answer without mentioning the keyword	1
	... correct keyword, but says the reason is something other than the word's presence	1
	... none of the above	0
Class imbalance	... folder contains more messages	2
	... folder size plus something else	1
	... folder contains more words	1
	... more predictions for the topic	1
	... none of the above	0

Table 6.1: The criteria used to grade participants' mental model questionnaire responses.

yield a mental model score with a maximum possible value of 24. The complete grading criteria are listed in Table 6.1.

We analyzed classifier performance via the F_1 *score*. This combines two simpler measures, *precision* and *recall*, each of which can range from 0 to 1. In the context of a binary classification system that predicts whether each input is positive or negative, a precision of 0 indicates that none of its positive predictions were correct, while a precision of 1 indicates that all of its positive predictions were correct. For the same system, a recall of 0 indicates the classifier did not correctly identify any of the positive items, while a recall of 1 indicates that it correctly identified all of them.

As the harmonic mean of precision and recall, F_1 also ranges from 0 (no precision and no recall) to 1 (perfect precision and recall). We calculate the F_1 score by identifying the number of *true positives* (items that are positive and that the classifier predicted as being positive), *false positive* (items that are negative but that the classifier predicted as being positive), *true negatives* (items that are negative and that the classifier predicted as being negative), and *false negatives* (items that are positive but that the classifier predicted as being negative):

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{6.1}$$

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{6.2}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.3}$$

A known problem with the F_1 score is that it ignores true negatives (Powers, 2011). As the above equations show, a classifier that has poor precision and recall for positive instances may perfectly predict negative instances, but its F_1 score will still be very low—the true negatives are not referenced by any of the equations. We can alleviate this issue by computing two F_1 scores: once as described above, then again after swapping the positive and negative instances. We then weight each score by the number of instances in its positive class and sum the results. For example, given two classes, *hockey* and *baseball*, we would compute an F_1 score with *hockey* as the positive class and weight it by the ratio of hockey instances to baseball instances. We would then compute a second F_1 score with *baseball* as the positive class, weighting the result by the ratio of baseball instances to hockey instances. Finally, we sum both scores together to yield a *weighted F_1 score* that will range between 0 (very poor precision and recall for all classes) and 1 (perfect precision and recall for all classes). All F_1 scores reported in this chapter are weighted scores.

We supplemented our evaluation of classifier performance with an additional offline experiment using a separate feature selection method. Recall that EluciDebug limits its classifier to the 10 features with the highest information gain. Text classifiers, however, often include most—if not all—of the words in the training set as features. Thus, we

analyzed participants' classifiers using both *HighIG* features and *Comprehensive* features. For control participants (who could not provide feature-based feedback), *HighIG* was recomputed after each message was labeled and kept the 10 highest information gain features. For treatment participants, *HighIG* was never recomputed; instead, participants needed to modify it manually by adding, removing, or adjusting features. The *Comprehensive* feature set included all words from the set of labeled messages at the end of the experiment. The classifiers participants interacted with used the *HighIG* features; the *Comprehensive* features were only used for offline analysis.

In addition to evaluating classifiers using the data that participants saw, we also used a validation set to verify that participants' classifiers were not overfit to their data. The *seen* dataset includes all of the messages that participants had the ability to view during the experiment; this includes messages that the participant labeled, as well as those that participants left unlabeled in the *Unknown* folder. The *unseen* dataset consists of a validation set that participants never saw (this is the *test* split of 20 Newsgroups). If the performance metrics show a difference between these two datasets, then we would have evidence that participants built classifiers overfitted to the data they interacted with during the experiment.

6.2 Results

6.2.1 Explaining corrections to EluciDebug (RQ6.1 and RQ6.2)

EluciDebug includes several methods for users to explain corrections to the classifier, and treatment participants made frequent use of all of them. These participants added new features, removed existing features, and adjusted feature importance; the average numbers for each action are shown in Table 6.2. Control participants—who could not provide feature-based feedback—instead relied on instance-based feedback to adjust EluciDebug's predictions, labeling nearly four times as many messages as treatment participants (Mann–Whitney U-test, $W = 1395$, $p < .001$) and examining nearly twice as many messages (Mann–Whitney U-test, $W = 1306$, $p < .001$). Treatment participants thus provided less feedback overall, and needed to explore less of the dataset while providing it. Instead, these participants leveraged EluciDebug's abilities to target their feedback at features rather than instances.

Action	Control mean (SD)	Treatment mean (SD)	*p*-value
Features added	–	34.5 (17.0)	–
Features removed	–	8.2 (3.3)	–
Features adjusted	–	18.3 (17.0)	–
Messages labeled	182.2 (91.8)	47.2 (46.4)	$< .001$
Message views	296.0 (111.2)	150.5 (78.0)	$< .001$

Table 6.2: The average usage of each of EluciDebug's feedback mechanisms. The *Features removed* total includes 7.4 of EluciDebug's 10 initial features. Overall, treatment participants targeted their feedback at features instead of instances.

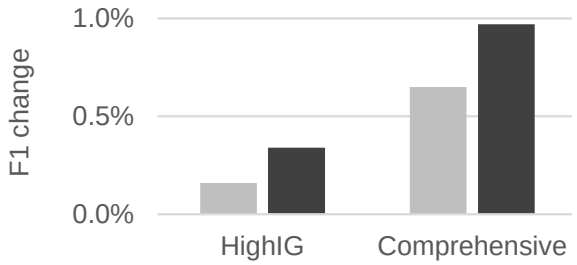


Figure 6.3: Average classifier F_1 improvement per user action for control (light) and treatment (dark); treatment participants controlled their classifiers up to twice as efficiently as control participants.

This feature-based feedback proved efficient at improving participants' classifiers. We examined the change in F_1 for each participant's classifier during the experiment and divided this by the number of actions the participant made that could influence the classifier's predictions (instances labeled and features added, removed, or adjusted). The results, shown in Figure 6.3, were that treatment participants performed fewer actions, but each of their actions resulted in larger classifier improvements than those of control participants. Treatment participants' feedback was twice as efficient as control participants' using *HighIG* features (0.16% vs. 0.34% F_1 improvement per action, Mann–Whitney U-test, $W = 207$, $p < .001$), and remained superior when using the *Comprehensive* feature set (0.65% vs. 0.97%, Mann–Whitney U-test, $W = 367$, $p < .001$).

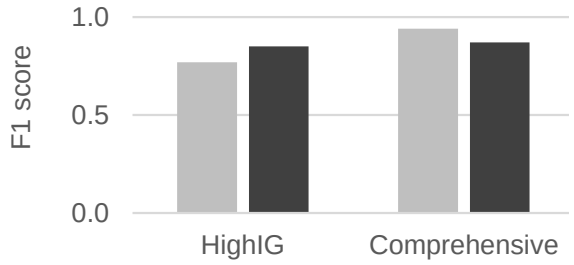


Figure 6.4: Average classifier F_1 scores per condition. Control participants needed four times as much data and the *Comprehensive* feature set (which included all words in the dataset as features) to create better classifiers than treatment participants.

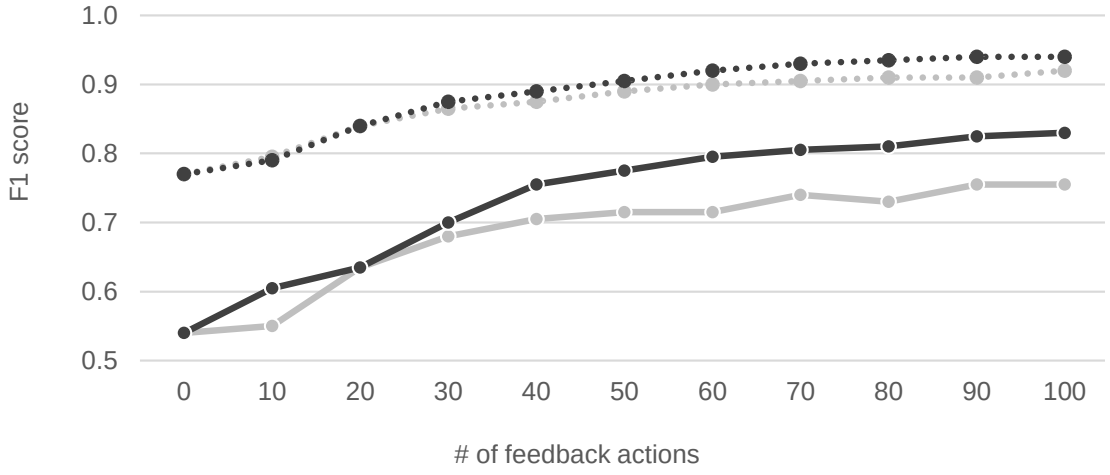


Figure 6.5: Treatment participants (dark) created equivalent or better classifiers than control participants (light) using the same amount of feedback. This held for both the *HighIG* (solid) and *Comprehensive* (dotted) feature sets.

We thus have evidence that when users can only provide a limited amount of feedback to a learning system (such as when labeling instances is expensive, insufficient instances are available for labeling, or the user's time is constrained), Explanatory Debugging can result in superior classifiers than a traditional black-box instance labeling approach. Indeed, Figure 6.4 shows that by the end of the 30-minute experiment, treatment participants had created classifiers that were roughly 10% more accurate than control participants, averaging F_1 scores of 0.85 vs. 0.77 (Mann–Whitney U-test, $W = 237$, $p < .001$).

However, our analysis with the *Comprehensive* feature set suggests that when the user can label many instances, instance labeling with a large feature set may be preferable to Explanatory Debugging—at least to initially train a classifier. The combination of a large training set and many features allowed control participants' classifiers to edge out those of treatment participants by about 8% (Figure 6.4). Even though treatment participants' feedback was up to twice as efficient, control participants provided almost four times as many labeled instances, allowing them to train classifiers with an average F_1 of 0.94, while treatment participants averaged 0.87 (Mann–Whitney U-test, $W = 1290$, $p < .001$).

To verify it was the amount of instance-based feedback that allowed control participants to outperform treatment participants when *Comprehensive* features were considered, we analyzed the accuracy of their classifiers after the same number of actions had been performed. Figure 6.5 shows the F_1 scores after n feedback actions using the *HighIG* (solid line) and *Comprehensive* (dotted line) feature sets. Given the same number of actions, control participants never outperformed treatment participants. This suggests that when treatment participants did provide instance-based feedback (which was the only type of feedback used for the *Comprehensive* analysis), it was usually more useful than control participants' feedback.

Table 6.3 presents the average F_1 scores for the four combinations of feature set and dataset we employed in our study and offline experiments. As the table shows, we found no evidence that participants overfit their classifiers to the data they saw during the study—their classifiers performed similarly on both the *seen* and *unseen* datasets.

We also analyzed participant reactions to the two prototype variations. Treatment participants liked their variant more than control participants, rating its helpfulness as 4.8 vs. 4.3 on a 6-point scale (Mann–Whitney U-test, $W = 474$, $p = .006$). Further, we did not find evidence that treatment participants felt Explanatory Debugging involved

Dataset	Feature set	Control F_1 (SD)	Treatment F_1 (SD)	p-value
Seen	HighIG	0.77 (0.07)	**0.85 (0.04)**	$< .001$
Unseen	HighIG	0.76 (0.06)	**0.84 (0.04)**	$< .001$
Seen	Comprehensive	**0.94 (0.03)**	0.87 (0.06)	$< .001$
Unseen	Comprehensive	**0.93 (0.03)**	0.86 (0.06)	$< .001$

Table 6.3: Average classifier accuracy for different datasets and feature selection methods. *Bold* denotes the significantly highest value between conditions.

more work than black-box instance labeling. We used the NASA-TLX survey to measure participants' perceived task load while attempting to improve their classifier, but found no evidence of a difference between conditions.

These classifier measures reveal three findings. First, in situations where large amounts of training data is unavailable or expensive to obtain, Explanatory Debugging (as instantiated in EluciDebug) allows users to successfully train a classifier by telling it about features instead of instances. Second, the mix of feature- and instance-based feedback provided by treatment participants was more efficient than the purely instance-based feedback provided by control participants, suggesting that when an end user has a specific goal in mind (such as our Alice example from Chapter 1), Explanatory Debugging can help the user quickly realize his or her goal.

Third, control participants' success with using large amounts of instance-based feedback suggests that in domains where labeling instances is quick and practical, some combination of feature- and instance-based feedback may be best. In fact, such systems may need to emphasize the potential usefulness of labeling instances. In our experiment, the mere presence of feature-based feedback tools appears to have biased participants against instance-based feedback: 3 treatment participants did not provide any at all, while the smallest number of labeled instances from a control participant was 56—more than even the treatment *average* of 47 labeled instances.

Model component	Max score	Control mean (SD)	Treatment mean (SD)	p-value
Obvious features	8	6.7 (2.7)	7.3 (1.8)	.345
Subtle features	8	2.8 (2.6)	**6.8 (1.9)**	<.001
Class ratios	8	0.6 (1.5)	1.8 (3.0)	.099
Total score	24	10.4 (5.3)	**15.8 (4.6)**	<.001

Table 6.4: Treatment participants finished the experiment with significantly higher mental model scores than control participants.

6.2.2 EluciDebug's explanations to end users (RQ6.3)

Before users can correct a machine learning system's explanation of its reasoning, they must first *understand* the explanation. This understanding is reflected in their mental model.

Treatment participants built significantly better mental models than participants in the control condition. As shown in Table 6.4, treatment participants scored 52% higher on the mental model assessment than control participants (Mann–Whitney U-test, $W = 259$, $p < .001$). Much of this difference stems from treatment participants identifying *all* of the keywords the classifier used, while control participants often identified only the "obvious" keywords. In fact, treatment participants averaged a score of 14.1 out of 16 (88%) during the keyword assessment, suggesting they firmly understood how the classifier involved keywords—regardless of whether the words had any semantic association with their topics—in its reasoning.

Table 6.4 also suggests treatment participants may have better understood that the classifier used class ratios as part of its prediction strategy than participants in the control condition (Mann–Whitney U-test, $W = 619.5$, $p = .099$), but the evidence is weak—even among treatment participants, the mean score was only 1.8 out of 8. Further, a majority of participants in both conditions failed to answer *any* class ratio question correctly, suggesting that this explanation either failed to convey relevant information about how class ratios were used by the classifier, or failed to attract participants' attention.

In general, however, control participants wanted the same information available to the treatment group. As one participant stated:

> *C1: "More information on how emails are sorted would help the user target emails to categorize, which would increase accuracy."*

Another control participant (C15) described the software as *"annoying"*, but that working with it *"would have been easier if we knew how it made predictions"*, while still another (C4) said it was annoying to work with because he *"didn't know what it was basing its predictions off of"*. A fourth participant (C11) even asked for a similar feedback mechanism as was available to treatment participants, saying the software was *"time-consuming"* because there was *"no way to highlight key words/terms"*. A fifth control participant succinctly summed up the entire experience:

> *C30: "That was a long 30 minutes."*

Participants in the treatment condition, conversely, voiced appreciation for EluciDebug's explanations and feedback mechanisms:

> *T24: "Not difficult to understand/operate, doesn't take a lot of effort."*

> *T40: "It was really fast to get a high degree of accuracy."*

> *T37: "I felt in control of all the settings."*

> *T6: "It was so simple my parents could use it."*

Overall, our principled Explanatory Debugging approach successfully helped participants develop accurate mental models of the classifier they used, and participants benefited from this additional knowledge. Spearman's ρ confirms a significant correlation between participants' mental model scores and their classifier's F_1 scores (Spearman's rank correlation coefficient, $\rho[75] = .282$, $p = .013$).

6.3 Discussion

6.3.1 Efficient and accurate personalization

Our results suggest that Explanatory Debugging can be an *efficient* method for users to personalize a machine learning system, but that it may not always result in the most *accurate* classifiers. For example, we found that feature-based feedback was up to twice as effective as instance-based feedback, but instance labeling could still yield more accurate classifiers given enough labels and features (in our experiment, four times as many labels were needed). In situations where labeling instances is considerably easier or faster than providing feature-based feedback, users may be better served by labeling a large number of instances than a small number of features.

However, when users need fine-grained control over a classifier, Explanatory Debugging has two advantages beyond efficiency. First, it does not require that training data exist, and thus can be used to bootstrap a learning system. Even a trained system that suddenly needs to support a new output type may benefit from such bootstrapping. Second, the quick improvements—during even the first 10 user actions—treatment participants made to their classifiers suggest that users will remain engaged with Explanatory Debugging. This matters because research has shown that if an end-user debugging technique is not perceived as useful after a small number of interactions, users are unlikely to continue using it (Prabhakararao et al., 2003). Seeing an immediate improvement after providing feedback suggests that users will continue to view and correct the Explanatory Debugging explanations, while the lack of such an improvement may discourage users of black-box instance labeling systems from continuing to provide feedback.

6.3.2 Mental models

Not only did Explanatory Debugging help participants build useful mental models, it accomplished this without a perceived increase in task load. Much as in our study of AuPair Radio in Chapter 3, we found no evidence that treatment participants found the extra information or feedback mechanisms more difficult to understand or use; instead, treatment participants' responses suggest they appreciated having such information available.

Indeed, the fact that many control participants' requested explanations remarkably similar to those the treatment participants saw suggests the need for machine learning systems to be able to explain their reasoning in accordance with Explanatory Debugging's *Explainability* principle. Even if this information is hidden by default, users should be able to view such explanations on demand. Further, because machine learning systems are meant to classify items as their user would, the user must have some method to correct the system's mistakes. Thus, we hypothesize that including Explanatory Debugging-style explanations without also supporting our *Correctibility* principle will frustrate users—they would be able to see what needs correcting, but without a clear mapping to the actions they need to take.

6.4 Conclusion

Overall, Explanatory Debugging's cycle of explanations—from the learning system to the user, and from the user back to the system—resulted in smarter users and smarter learning systems. Participants using Explanatory Debugging understood how the learning system operated about 50% better than control participants, and this improvement was positively correlated with the F_1 scores of participants' classifiers. Each piece of feedback provided by Explanatory Debugging participants was worth roughly two pieces of feedback provided by control participants; even when we expanded our analysis to include a comprehensive feature set, treatment participants still maintained a 50% efficiency edge over the control group. Further, participants *liked* Explanatory Debugging, rating this variant of EluciDebug higher than the control group and responding enthusiastically to the system's explanations.

Chapter 7: Conclusion

Without personalization, classifiers and recommenders are simply static tools, incapable of adapting to new inputs or a user's shifting expectations of its outputs. Spam filters would not adapt to new types of junk mail. Netflix would not detect your shifting taste is movies. Banks would have difficulty identifying anomalous activity on your credit cards as your spending habits change.

Even when they support personalization, however, many machine learning systems treat it as something that happens in the background. Users provide the system with new labeled instances (e.g., by moving email to the junk folder, or by rating a movie on Netflix), but they are not told how the system has updated itself in response to this new information. This dissertation has argued that users need more fine-grained control in situations where this traditional instance-based approach breaks down—when personalization must happen quickly, when an insufficient amount of training data exists, when the user's concept has evolved, or when the user needs a better understanding of how the learning system makes its predictions.

As this dissertation has shown, a key concept of helping end users personalize a machine learning system is to help them understand how it operates. Our research found that as users improved their mental models of a learning system, they were better able to personalize it to suit their needs. Moreover, we discovered that helping users build high-fidelity mental models is surprisingly feasible, and that end users were not intimidated or put-off by explanations of how the learning system operates "under the hood". Instead, high-fidelity mental models were tied to increased computer self-efficacy, which suggests that these users will be more perseverant than users holding low-fidelity mental models if they encounter barriers while personalizing their learning systems.

Helping users develop high-fidelity mental models is challenging because machine learning systems can use a dizzying amount of material to reach their predictions and recommendations. We had initially hypothesized that machine-generated explanations of this process would need to be somewhat abstracted—but not too much—in order to help end users build useful mental models. However, our qualitative investigation of

explanation fidelity suggested that explanations should be both as sound and complete as possible in order to help users build high-fidelity mental models; that very complete explanations can help improve the perceived cost/benefit trade-off of attending to explanations; and that very sound explanations were perceived as more trustworthy than very unsound explanations. Our research also revealed that Lim and Dey's *why*, *input*, and (especially) *model* intelligibility types were often referenced when participants made correct statements about the learning system's reasoning, while the *certainty* intelligibility type was largely ignored. These intelligibility types provide a systematic method for increasing the completeness of a system's explanations.

Given these promising results, we developed a new approach for controlling machine learning systems. We used the findings from our investigations of mental models and explanation fidelity to inform the design of Explanatory Debugging, an explanation-centric approach to help end users efficiently personalize their learning systems. Explanatory Debugging is based upon a two-way cycle of interactive, context-sensitive explanations: the learning system explains the reasons underlying its predictions or recommendations to the user, who can then explain any necessary corrections back to the system. Our empirical evaluation of Explanatory Debugging found that both halves of this cycle worked as intended—users of our Explanatory Debugging prototype developed significantly better mental models than users of a traditional black-box learning system, and they also personalized their learning systems' reasoning significantly more efficiently. Further, users reacted more enthusiastically to Explanatory Debugging than to a traditional machine learning system, and we found no evidence that personalization via Explanatory Debugging—with its high-fidelity explanations—was perceived as more mentally taxing than trying to understand and personalize a black-box learning system.

Together, the results of this dissertation show that when end users want to personalize a machine learning system, Explanatory Debugging is a more controllable and satisfying approach than black-box instance labeling. Our approach's focus on users—in which the system explains its reasoning and the user explains back corrections as needed—enables everyday users to get the most out of the learning systems upon which they are beginning to depend.

Bibliography

Abraham, R. and Erwig, M. (2006). AutoTest: A tool for automatic test case generation in spreadsheets. In *Proceedings of the IEEE 2006 Symposium on Visual Languages and Human-Centric Computing*, pages 43–50.

Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. (2009). Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, pages 247–256.

Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. (2010). Examining multiple potential models in end-user interactive concept learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1357–1360.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2):191–215.

Becker, B., Kohavi, R., and Sommerfield, D. (2001). Visualizing the simple Baysian classifier. In Fayyad, U., Grinstein, G. G., and Wierse, A., editors, *Information Visualization in Data Mining and Knowledge Discovery*, pages 237–249. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Benedek, J. and Miner, T. (2002). Measuring desirability: New methods for evaluating desirability in a usability lab setting. In *Proceedings of the Usability Professionals' Association Conference 2002*, Orlando, FL, USA.

Billsus, D., Hilbert, D. M., and Maynes-Aminzade, D. (2005). Improving proactive information systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 159–166.

Blackwell, A. F. (2002). First steps in programming: A rationale for attention investment models. In *Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments*, pages 2–10.

Bostandjiev, S., O'Donovan, J., and Höllerer, T. (2012). TasteWeights: A visual interactive hybrid recommender system. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 35–42.

Bostandjiev, S., O'Donovan, J., and Höllerer, T. (2013). LinkedVis: Exploring social and semantic career recommendations. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, pages 107–115.

Bozionelos, N. (2001). The relationship of instrumental and expressive traits with computer anxiety. *Personality and Individual Differences*, 31:955–974.

Brain, D. and Webb, G. (1999). On the effect of data set size on bias and variance in classification learning. In *Proceedings of the Fourth Australian Knowledge Acquisition Workshop*, pages 117–128.

Bryan, N. J., Mysore, G. J., and Wang, G. (2014). ISSE: An interactive source separation editor. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 257–266.

Bunt, A., Lount, M., and Lauzon, C. (2012). Are explanations always important? A study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 International Conference on Intelligent User Interfaces*, pages 169–178.

Cakmak, M., Chao, C., and Thomaz, A. L. (2010). Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118.

Carley, K. and Palmquist, M. (1992). Extracting, representing, and analyzing mental models. *Social Forces*, 70(3):601–636.

Carroll, J. M. and Rosson, M. B. (1987). Paradox of the active user. In Carroll, J. M., editor, *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, pages 80–111. MIT Press, Cambridge, MA, USA.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3).

Compeau, D. R. and Higgins, C. A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly*, 19(2):189–211.

Corbin, J. M. and Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology*, 13(1):3–21.

Cramer, H., Evers, V., Ramlal, S., van Someren, M., Rutledge, L., Stash, N., Aroyo, L., and Wielinga, B. (2008). The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-Adapted Interaction*, 18(5):455–496.

Craven, M. W. and Shavlik, J. W. (1997). Using neural networks for data mining. *Future Generation Computer Systems*, 13:211–229.

Das, S., Moore, T., Wong, W.-K., Stumpf, S., Oberst, I., McIntosh, K., and Burnett, M. M. (2013). End-user feature labeling: Supervised and semi-supervised approaches based on locally-weighted logistic regression. *Artificial Intelligence*, 204:56–74.

Doyle, J. K., Radzicki, M. J., and Trees, W. S. (2008). Measuring change in mental models of complex dynamic systems. In *Complex Decision Making*, pages 269–294. Springer Berlin Heidelberg, Berlin, Heidelberg.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H. P. (2003). The role of trust in automation reliance. *International Journal of Human-Computer Studies*, 58(6):697–718.

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

Fiebrink, R., Cook, P. R., and Trueman, D. (2011). Human model evaluation in interactive supervised learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 147–156.

Fisher II, M., Rothermel, G., Brown, D., Cao, M., Cook, C., and Burnett, M. M. (2006). Integrating automated test generation into the WYSIWYT spreadsheet testing methodology. *Transactions on Software Engineering and Methodology*, 15(2):150–194.

Fogarty, J., Tan, D., Kapoor, A., and Winder, S. (2008). CueFlik: Interactive concept learning in image search. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 29–38.

Glass, A., McGuinness, D. L., and Wolverton, M. (2008). Toward establishing trust in adaptive agents. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 227–236.

Glowacka, D., Ruotsalo, T., Konuyshkova, K., Athukorala, k., Kaski, S., and Jacucci, G. (2013). Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, pages 117–127.

Groce, A., Kulesza, T., Zhang, C., Shamasunder, S., Burnett, M. M., Wong, W.-K., Stumpf, S., Das, S., Shinsel, A., Bice, F., and McIntosh, K. (2014). You are the only possible oracle: Effective test selection for end users of interactive machine learning systems. *IEEE Transactions on Software Engineering*, 40(3):307–323.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., and Reutemann, P. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.

Hart, S. G. and Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in Psychology*, 52:139–183.

Hastie, R. (1984). Causes and effects of causal attribution. *Journal of Personality and Social Psychology*, 46(1):44–56.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, volume 2. Springer, New York, NY, USA.

Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250.

Jakulin, A., Možina, M., Demšar, J., Bratko, I., and Zupan, B. (2005). Nomograms for visualizing support vector machines. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 108–117.

Johnson-Laird, P. N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard University Press.

Jonassen, D. H. and Henning, P. (1996). Mental models: Knowledge in the head and knowledge in the world. In *Proceedings of the 1996 International Conference on Learning Sciences*, pages 433–438.

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

Kempton, W. (1986). Two theories of home heat control. *Cognitive Science*, 10:75–90.

Kibriya, A. M., Frank, E., Pfahringer, B., and Holmes, G. (2004). Multinomial naive Bayes for text categorization revisited. In *AI 2004: Advances in Artificial Intelligence*, pages 488–499. Springer Berlin Heidelberg, Berlin, Heidelberg.

Knox, W. B. and Stone, P. (2012). Reinforcement learning from human reward: Discounting in episodic tasks. In *Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 878–885.

Ko, A. J. (2006). Debugging by asking questions about program output. In *Proceeding of the 28th International Conference on Software Engineering*, pages 989–992.

Ko, A. J. and Myers, B. A. (2008). Debugging reinvented: Asking and answering why and why not questions about program behavior. In *Proceedings of the 13th International Conference on Software Engineering*, pages 301–310.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, NJ, USA.

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

Kulesza, T., Stumpf, S., Burnett, M. M., Wong, W.-K., Riche, Y., Moore, T., Oberst, I., Shinsel, A., and McIntosh, K. (2010). Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 41–48.

Kulesza, T., Stumpf, S., Wong, W.-K., Burnett, M. M., Perona, S., Ko, A. J., and Oberst, I. (2011). Why-oriented end-user debugging of naive Bayes text classification. *ACM Transactions on Interactive Intelligent Systems*, 1(1).

Lacave, C. and Díez, F. J. (2002). A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127.

Lieberman, H. (2001). *Your Wish is My Command: Programming by Example*. Morgan Kaufmann.

Lim, B. Y. (2012). *Improving understanding and trust with intelligibility in context-aware applications*. PhD thesis, Carnegie Mellon University.

Lim, B. Y. and Dey, A. K. (2009). Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, pages 195–204.

Lim, B. Y., Dey, A. K., and Avrahami, D. (2009). Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 2119–2128.

McCrum-Gardner, E. (2008). Which is the correct statistical test to use? *British Journal of Oral and Maxillofacial Surgery*, 46:38–41.

McDaniel, R. G. and Myers, B. A. (1997). Gamut: Demonstrating whole applications. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, pages 81–82.

McNee, S. M., Lam, S. K., Guetzlaff, C., Konstan, J. A., and Riedl, J. (2003). Confidence displays and training in recommender systems. In *Proceedings of INTERACT '03*, pages 176–183.

Miller, R. C. and Myers, B. A. (2001). Outlier finding: Focusing user attention on possible errors. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, pages 81–90.

Možina, M., Demšar, J., Kattan, M., and Zupan, B. (2004). Nomograms for visualization of naive Bayesian classifier. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Knowledge Discovery in Databases: PKDD 2004*, pages 337–348. Springer Berlin Heidelberg, Berlin, Heidelberg.

Myers, B. A., Weitzman, D. A., Ko, A. J., and Chau, D. H. (2006). Answering why and why not questions in user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 397–406.

Norman, D. A. (1987). Some observations on mental models. In Baecker, R. M. and Buxton, W. A. S., editors, *Human-Computer Interaction*, pages 241–244. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Norman, D. A. (2002). *The Design of Everyday Things*. Basic Books, New York, NY, USA.

Otter, M. and Johnson, H. (2000). Lost in hyperspace: Metrics and mental models. *Interacting with computers*, 13:1–40.

Parra, D., Brusilovsky, P., and Trattner, C. (2014). See what you want to see: Visual user-driven approach for hybrid recommendation. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*, pages 235–240.

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D. S., Fyshe, A., Pearcy, B., MacDonell, C., and Anvik, J. (2006). Visual explanation of evidence with additive classifiers. In *Proceedings of the Fourth Australian Knowledge Acquisition Workshop*, pages 1822–1829.

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, Informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Prabhakararao, S., Cook, C., Ruthruff, J., Creswick, E., Main, M., Durham, M., and Burnett, M. M. (2003). Strategies and behaviors of end-user programmers with interactive fault localization. In *Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments*, pages 15–22.

Raghavan, H., Madani, O., and Jones, R. (2006). Active learning with feedback on features and instances. *The Journal of Machine Learning Research*, 7:1655–1686.

Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*.

Raz, O., Koopman, P., and Shaw, M. (2002). Semantic anomaly detection in online data sources. In *Proceedings of the 24th International Conference on Software Engineering*, pages 302–312.

Rogers, Y., Sharp, H., and Preece, J. (2011). *Interaction Design: Beyond Human - Computer Interaction*. John Wiley & Sons.

Rosson, M. B., Carrol, J. M., and Bellamy, R. K. E. (1990). Smalltalk scaffolding: A case study of minimalist instruction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 423–429.

Rothermel, G., Burnett, M. M., Li, L., Dupuis, C., and Sheretov, A. (2001). A methodology for testing spreadsheets. *ACM Transactions on Software Engineering and Methodology*, 10(1):110–147.

Rowe, M. B. (1973). *Teaching Science as Continuous Inquiry*. McGraw-Hill.

Scaffidi, C. (2007). Unsupervised Inference of Data Formats in Human-Readable Notation. In *Proceedings of 9th International Conference on Enterprise Integration Systems*, pages 236–241.

Settles, B. (2010). Active learning literature survey. Technical Report 1648, University of Wisconsin-Madison.

Shneiderman, B. and Plaisant, C. (2010). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 5th edition.

Sinha, R. and Swearingen, K. (2002). The role of transparency in recommender systems. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 830–831.

Stone, P. J. (1968). *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Storey, M., Fracchia, F. D., and Müller, H. A. (1999). Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Systems and Software*, 44:171–185.

Stumpf, S., Rajaram, V., Li, L., Burnett, M. M., Dietterich, T., Sullivan, E., Drummond, R., and Herlocker, J. (2007). Toward harnessing user feedback for machine learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, pages 82–91.

Stumpf, S., Rajaram, V., Li, L., Wong, W.-K., Burnett, M. M., Dietterich, T., Sullivan, E., and Herlocker, J. (2009). Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639–662.

Stumpf, S., Sullivan, E., Fitzhenry, E., Oberst, I., Wong, W.-K., and Burnett, M. M. (2008). Integrating rich user feedback into intelligent user interfaces. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 50–59.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

Szafron, D., Greiner, R., Lu, P., and Wishart, D. (2003). Explaining naïve Bayes classifications. Technical Report TR03-09, University of Alberta.

Talbot, J., Lee, B., Kapoor, A., and Tan, D. S. (2009). EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1283–1292.

Thomaz, A. L. and Breazeal, C. (2006). Transparency and socially guided machine learning. In *Proceedings of the 5th International Conference on Development and Learning*.

Tintarev, N. and Masthoff, J. (2012). Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):399–439.

Tullio, J., Dey, A. K., Chalecki, J., and Fogarty, J. (2007). How it works: A field study of non-technical users interacting with an intelligent system. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 31–40.

van der Meij, H. and Carroll, J. M. (1998). Principles and heuristics for designing minimalist instruction. In Carroll, J. M., editor, *Minimalism Beyond the Nurnberg Funnel*, pages 19–53. MIT Press, Cambridge, MA.

Verbert, K., Parra, D., Brusilovsky, P., and Duval, E. (2013). Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, pages 351–362.

Vig, J., Sen, S., and Riedl, J. (2011). Navigating the tag genome. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 93–102.

Wagner, E. J. and Lieberman, H. (2004). Supporting user hypotheses in problem diagnosis. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 30–37.

Wilfong, J. D. (2006). Computer anxiety and anger: the impact of computer use, computer experience, and self-efficacy beliefs. *Computers in Human Behavior*, 22:1001–1011.

Wilson, A., Burnett, M. M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., Durham, M., and Rothermel, G. (2003). Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 305–312.

Yang, R. and Newman, M. W. (2013). Learning from a learning thermostat: Lessons for intelligent systems for the home. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 93–102.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 412–420.

APPENDICES

Appendix A: ElluciDebug study materials

This appendix presents the materials used in our study of ElluciDebug (Chapter 6). The first five pages present the tutorial we used to introduce participants to ElluciDebug. Text highlighted in green explains what participants saw via a projector. Text highlighted in blue was only provided to treatment participants, while text highlighted in salmon was only provided to control participants.

We next present the background, pre-task, and post-task questionnaires. Because treatment participants used a different variant of ElluciDebug than control participants, there are condition-specific variants of the pre- and post-task questionnaires. For each questionnaire, we present the control variant before the treatment variant.

Finally, we also include the answer keys for our mental model assessments. These answers list either the specific keyword responsible for the classifier's prediction, or "class imbalance" if the prediction was determined by class ratios instead of keyword presence.

Message Predictor Tutorial

Hi, my name is [your name], and I'll be walking you through our study. [Introduce the rest of the study team]. If your cell phone is on, please set it to silent mode.

I'll be reading from this script to be consistent in the information I provide you and the other people taking part in this study. Please don't discuss this study with anyone, as we don't want other participants to receive any advance information. During this tutorial, please follow along with my directions and don't explore the system by yourself yet.

This study is about working with software that makes predictions. For example, most email software makes predictions about whether each new message is important, or junk mail. The software will typically learn—based on your behavior—which types of messages you think are important, and which you think are junk.

Today I'll show you a tool that tries to predict whether new messages are about one of two topics. For this tutorial, the topics will be *medicine* and *outer space*. The messages that we'll look at were collected from Internet forums in 1994, so they're real messages, but way out of date. Let's click "OK" to start the application.

[Click "OK" and wait for the application to start]

Alright, let's go over the different parts.

In the upper-left, there's a list of folders [Hover mouse around folder list]. There are only three, and you can't change them. The first is "Unknown", which has over 1,000 messages in it. Those messages are shown in the middle of the screen. The selected message is shown on the right side of the screen.

The second folder is "Medicine" and has 10 messages in it. Let's select that folder to view those messages [Click on Medicine folder]. Notice that this folder tells us that currently all 10 messages are correctly predicted as being about "medicine" [Hover around "10/10 correct predictions"].

The third folder is "Space" and also has 10 messages in it. Let's select it [Click on Space folder]. This folder tells us that only 8 of the 10 messages are correctly predicted. If we look at the "Predicted topic" column in the message list, we can see which messages the computer is wrongly predicting; their predicted topic is "Medicine" instead of "Space" [Hover mouse around incorrect predictions]. Notice that these topics are color-coded, with teal for "space" and red for "medicine".

Now let's go back to the "Unknown" folder [Click on Unknown folder]. These message all have a predicted topic as well [Hover mouse around "Predicted topic" column]. We can sort the messages by their predicted topic by clicking the column header, like this [Click "Predicted topic" header to sort messages]. Now all of the messages that the computer thinks are about medicine are at the top of the list. If we scroll down, we'll see

that all of the messages the computer thinks are about space are at the bottom [Scroll down to the very bottom of the list].

We can also sort the messages by Subject. Let's click on the "Subject" column header to do that [Click "Subject" header to sort messages].

There's also a column showing how confident the computer is about its prediction [Hover mouse around Prediction Confidence column]. This can range from 99%, which means the computer is very confident in its prediction, to 50%, which means the computer is just guessing. Just like the other columns, we can also sort messages by confidence. Let's do that now [Click "Prediction Confidence" header to sort messages]. This puts all of the very unconfident messages at the top of the list, and the most confident messages at the bottom [Scroll down to the very bottom of the list].

If you want to return to the original sort order, you can do that by clicking the "Original order" column. Let's do that now [Click "Order" header to sort messages]. Let's also scroll all the way back to the top of the list [Scroll to top of message list].

Now this is really important. The predictions that the computer makes *may not be correct*. In fact, the purpose of this software is to try to improve the computer's predictions by giving it more information. You can do that / One way to do that is by moving messages to the "Medicine" or "Space" folder, which will tell the computer to learn from those messages. To see how that works, let's select the "NASA Wraps" message [Select message].

With a word like "NASA" in the subject, it's probably a safe bet that this message is about outer space. If you weren't certain, you can always read as much—or as little—of the message as you'd like. If you think the computer should learn more about a topic from this message, we can move it to either the "Space" or "Medicine" folder. To do that, make sure the message is selected and click the "Move message to folder…" button [Click button and hover over "space", but don't select it].

You can use this button to move a message into any folder. You're free to move as many, or as few, messages into each folder as you like. You don't need to move every message into a folder. For now, let's go ahead and move this message into the "Space" folder [Move message to "space"].

Now that we've told the computer to learn from this message, a few things have changed. First, you may have noticed that the predicted topic for many messages in the Unknown folder changed from medicine to space, or vice versa [Hover around the "Predicted topic" column]. When the computer changes its prediction for a message, the predicted topic will have a grey background, like these. Second, the number of correct predictions for Medicine and Space have changed [Hover around Medicine and Space folders]. Medicine used to have 10 correct predictions, while Space only had 8. Now all of the Space predictions are correct, but one Medicine prediction is wrong / four of the Medicine

predictions are wrong. These green "up" arrows and red "down" arrows tell you whether a number just increased or decreased; you can hover over them to see the exact change [Mouse over the red arrow next to "Medicine" until the tooltip appears]. Below the Folder List you can see the total number of messages the computer predicts are about each topic [Hover mouse around Prediction Totals]. Here we see that the computer currently predicts that 179 messages are about medicine, while over 1,000 messages are about space. Adding that one message to the Space folder helped the computer improve its predictions about Space, but at the cost of its predictions about Medicine.

So, moving messages into the Medicine or Space folders may help you improve the computer's predictions, and it also allows you to see how accurate those predictions are. There's no limit on how many messages may be in each folder.

Another way for you to improve the computer's predictions is to adjust the Important Words that it pays attention to. These are listed at the bottom of the screen [Hover mouse around Important Words section]. Each of these words will also be highlighted in blue in messages [Hover mouse around highlighted words in message]. You can add new words or phrases, remove existing words, or adjust the relative importance of words.

For practice, let's click on "Steve" [Click "Steve" in important words list]. When we select a word, we get an overview of all the messages that contain this word in the lower-left of the screen [Hover mouse around Messages Containing Steve section]. This shows that "Steve" occurs in 3 of the 10 messages in our Medicine folder [Hover mouse around Medicine in heatmap], but none of the messages in our Space folder [Hover mouse around Space in heatmap], and another few dozen messages that are still in the Unknown folder [Hover mouse around Unknown in heatmap].

My first thought is that the word "Steve" probably doesn't have much to do with medicine, but we can click on each message to see how the word is used. Let's click the first highlighted message in Medicine [Click first message in medicine heatmap]. The word will be highlighted in light blue, but we may need to scroll through the message to find it [Scroll down to end of message]. So this message was sent by someone named Steve. Let's look at the next two messages [Click on the next two messages in the medicine heatmap, slowly enough for participants to see that Steve is the name of the person who sent each message]. It looks like the computer noticed that someone named Steve sent three messages about Medicine, and no one named Steve sent any messages about Space. Let's click "Close" to go back to the main window [Close dialog window].

I can't think of a reason why Steve should be more important to Medicine than Outer Space, so let's remove "Steve" as an important word [Click on "Remove Steve" button]. Checkout what happened to the Medicine folder—its accuracy dropped down to 5 out of 10 messages [Hover over down arrow next to Medicine]. Let's click on that folder to look at some of its messages [Click on the Medicine folder]. So this first message has the subject "Too many MRIs". "MRIs" seems like it might be an important word about

medicine. Let's click "Add a new word or phrase" to add it [Click "Add a new word or phrase" button].

We'll type in "MRIs" [Slowly type "MRIs"]. Notice that we include the 'S' on the end, and that as we type it, we can see it highlight in the message. If we don't include the 'S', it won't highlight. The computer doesn't know about differences in pluralization or spelling, so if you want it to pay attention to a certain word or phrase, you need to enter that word *exactly* as you see it in the message. Also notice that the message overview in the lower-left shows up, so we can preview other messages that contain this word.

Now we need to select the topic associated with this word; we'll pick Medicine and then click "Add" [Select Medicine from the menu and click the "Add" button].

Besides adding or removing words and phrases, you can also adjust their importance. For example, let's select "Gravity" from the list of important words [Click on "Gravity"]. The teal bar shows how important this word is for predicting messages about Space, and the red bar shows how important this word is for predicting messages about Medicine. We can drag each bar up to make it more important, or down to make it less important. However, if there's a black line on the bar, we can't drag it any lower than that line. For example, if we wanted to tell the computer that "Gravity" should be more important to Medicine than Space, we could drag the teal bar down, but it won't let us drag it below that line [Drag the teal bar as low as it will go]. Instead, we could drag the red bar up to tell the computer that Gravity is more important to medicine than space [Drag the red bar about twice as high as the teal bar].

The thing is, "gravity" probably should be associated with space more than medicine, so let's undo our changes [Clicks "Undo importance adjustment"]. In fact, "Gravity" should probably be much more strongly associated with space than medicine, so let's drag it up [Drag teal bar up to the level of "Moon"].

As we've been working with this software, you've probably noticed the animations on the right side of the screen [Hover mouse around "Why medicine?" explanation]. This area explains why the computer is making each of its predictions; each message has its own explanation. When you adjust important words or move messages into different folders, you might change the computer's reasons for some of its predictions. When the computer's reasoning changes, this explanation will animate to let you know that it's changed.

Finally, there are a few features designed to help you work faster with this software. You can select words and phrases in message and right-click to either find all of the messages containing the word, or add it as an important word or phrase [Select some text and right-click to show the context menu]. You can also search for any word you want using this Search box [hover mouse around "Search" box]. The results will show up in the lower-left, just like when you add or select Important Words [Type "test" into the Search box, the hover mouse around the heatmap]. If you want to focus on predictions that were

altered by your last action, you can use the "Only show predictions that just changed" button [Hover mouse around "Only show…" button]. We can turn that on right now [Click "Only show…" button to enable it], but since none of the messages in this folder recently changed their prediction, nothing shows up. So let's turn that option back off for now [Click "Only show…" button to disable it].

I'm going to give you a couple of minutes to practice using this software, and then we'll start the study. Let one of us know if you have any questions.

[Wait for three minutes.]

Alright, now please close the software you've been working with. You should see a small window appear, but don't click "OK" yet.

Before we start the study, we have some questions we'd like you to answer. These questions ask you to imagine working with the same software you just saw, but instead of messages about medicine and outer space, the messages are about soccer and basketball. Answer the questions as best you can, with as much detail as possible. You'll have about 10 minutes.

[Pass out pre-task questions and wait.]

Alright, now we'll start the main part of the study. You'll be using the same software you saw during the tutorial, but instead of messages about medicine and outer space, the messages will be about baseball and ice hockey. The colors associated with those topics may also be different. Everything else about the software is still the same.

Your task is to make the computer's predictions as accurate as possible. You'll have 30 minutes, and may begin now by clicking "OK".

[Wait 30 minutes.]

Alright, time's up! Go ahead and click "Close" to close the application. Before we finish, we have some final questions we'd like you to answer. You'll have up to 20 minutes. Let us know once you're done and we'll pay you for helping us with this research. Thanks for coming in today!

Group: _____ Participant: _____

Background questions

Age _____ years

Gender (circle one) *Female* *Male*

College standing (circle one)
Undergraduate
Graduate student
Already graduated
Not a student

College major _____

College GPA (approximate) _____

I know the rules of professional baseball

Strongly disagree *Strongly agree*

I pay attention to professional baseball

Strongly disagree *Strongly agree*

I know the rules of professional ice hockey

Strongly disagree *Strongly agree*

I pay attention to professional ice hockey

Strongly disagree *Strongly agree*

Group: _____ Participant: _____

This part of the questionnaire asks you about your ability to use an unfamiliar piece of software. Often in our jobs we are told about software that is available to make our work easier. For the following questions, imagine that you were given a new software package for some aspect of your work. It doesn't matter specifically what this software does, only that it is intended to make your job easier and that you have never used it before.

The following questions ask you to indicate whether you could use this unfamiliar software under a variety of conditions. For each of the conditions, indicate whether you think you would be able to complete the job using the software. Then, for each condition that you answered "YES", please rate your confidence about your first judgment by circling a number from 1 to 10, where 1 indicates "Not at all confident", 5 indicates "Moderately confident", and 10 indicates "Totally confident".

I could complete my job using this software…

		Not at all confident					Moderately confident				Totally confident
…if there was no one around to tell me what to do as I go.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had never used a package like it before.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had only the software manuals for reference.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had seen someone else using it before trying it myself.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I could call someone for help if I got stuck.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if someone else had helped me get started.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had a lot of time to complete the job for which the software was provided.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had just the built-in help facility for assistance.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if someone showed me how to do it first.	YES… NO	1	2	3	4	5	6	7	8	9	10
…if I had used similar packages before this one to do the same job.	YES… NO	1	2	3	4	5	6	7	8	9	10

Group: _____ Participant: _____

Pre-task questions

Assume you are using the same software you just worked with, but instead of *Medicine* and *Space*, it predicts whether each message is about *Soccer* or *Basketball*. The software has learned from two messages in the *Soccer* folder, and one message in the *Basketball* folder:

Soccer messages:

#1 **Subject: Soccer is the best**
Soccer is truly the best sport in the world!

#2 **Subject: Soccer is the best**
Agreed!

Basketball message:

#1 **Subject: Basketball is the best**
No, I think basketball is the best sport in the world!

Assume that **these are the only messages** the software has learned from. The following page lists several messages in the *Unknown* folder. For each message, circle the topic that you think the computer will predict and explain why. Be as specific as possible.

Group: _____ Participant: _____

Message in *Unknown* folder	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules Truly, rugby is the best sport!	Soccer	Basketball	
Subject: Basketball does rule Right on!	Soccer	Basketball	
Subject: Rugby rules I think rugby is best.	Soccer	Basketball	
Subject: What?! Basketball beats soccer any day.	Soccer	Basketball	
Subject: Soccer does rule Right on!	Soccer	Basketball	
Subject: Soccer forever It's just more fun than basketball.	Soccer	Basketball	
Subject: Yeah! Agreed, rugby rocks.	Soccer	Basketball	
Subject: Nope! No, rugby does not rock.	Soccer	Basketball	
Subject: Chess is better You're all wrong.	Soccer	Basketball	
Subject: Eh? Chess is more exciting.	Soccer	Basketball	
Subject: Pshhhh But soccer is more popular.	Soccer	Basketball	
Subject: Not in the states Basketball is more popular here.	Soccer	Basketball	

Pre-task questions

Group: _____ Participant: _____

Please list all of the words you can think of that are about Hockey:

Please list all of the words you can think of that are about Baseball:

Pre-task questions

Group: _____ Participant: _____

Pre-task questions

Assume you are using the same software you just worked with, but instead of *Medicine* and *Space*, it predicts whether each message is about *Soccer* or *Basketball*. The software has learned from two messages in the *Soccer* folder, and one message in the *Basketball* folder:

Soccer messages:

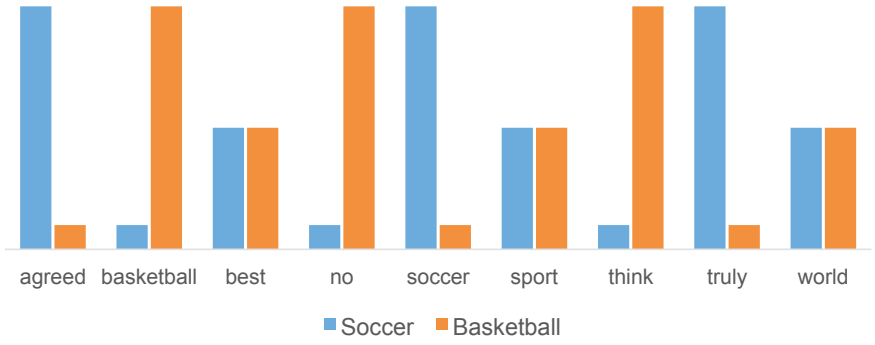
#1 **Subject: Soccer is the best**
Soccer is truly the best sport in the world!

#2 **Subject: Soccer is the best**
Agreed!

Basketball message:

#1 **Subject: Basketball is the best**
No, I think basketball is the best sport in the world!

The **Important Words** section of the software shows the following:



Assume that **these are the only messages** the software has learned from. The following page lists several messages in the *Unknown* folder. For each message, circle the topic that you think the computer will predict and explain why. Be as specific as possible.

Group: _____ Participant: _____

Message in *Unknown* folder	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules Truly, rugby is the best sport!	Soccer	Basketball	
Subject: Basketball does rule Right on!	Soccer	Basketball	
Subject: Rugby rules I think rugby is best.	Soccer	Basketball	
Subject: What?! Basketball beats soccer any day.	Soccer	Basketball	
Subject: Soccer does rule Right on!	Soccer	Basketball	
Subject: Soccer forever It's just more fun than basketball.	Soccer	Basketball	
Subject: Yeah! Agreed, rugby rocks.	Soccer	Basketball	
Subject: Nope! No, rugby does not rock.	Soccer	Basketball	
Subject: Chess is better You're all wrong.	Soccer	Basketball	
Subject: Eh? Chess is more exciting.	Soccer	Basketball	
Subject: Pshhhh But soccer is more popular.	Soccer	Basketball	
Subject: Not in the states Basketball is more popular here.	Soccer	Basketball	

Pre-task questions

Group: _____ Participant: _____

Please list all of the words you can think of that are about Hockey:

Please list all of the words you can think of that are about Baseball:

Pre-task questions

Group: _____ Participant: _____

Post-task questions

Assume you are using the same software you just worked with, but instead of *Baseball* and *Hockey*, it predicts whether each message is about *Swimming* or *Tennis*. The software has learned from two messages in the *Swimming* folder, and one message in the *Tennis* folder:

Swimming messages:

#1 **Subject: Swimming is the worst**
Swimming is certainly the worst sport in the world.

#2 **Subject: Swimming is the worst**
Seriously?

Tennis message:

#1 **Subject: Tennis is the worst**
Wrong, tennis is totally the worst sport in the world.

Assume that **these are the only messages** the software has learned from. The following page lists several messages in the *Unknown* folder. For each message, circle the topic that you think the computer will predict and explain why. Be as specific as possible.

Group: _____ Participant: _____

Message in *Unknown* folder	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules! You're wrong, rugby's awesome!	Swimming	Tennis	
Subject: Meh Chess is more exciting.	Swimming	Tennis	
Subject: Swimming rules Well it does.	Swimming	Tennis	
Subject: Yeah! Rugby totally rocks.	Swimming	Tennis	
Subject: Pshhhh But tennis is more popular.	Swimming	Tennis	
Subject: Rugby is dull Rugby is certainly the worst sport.	Swimming	Tennis	
Subject: Tennis rules Tennis is best.	Swimming	Tennis	
Subject: Chess can be exciting If you wear a helmet.	Swimming	Tennis	
Subject: Nope! Seriously, helmets do not make chess exciting.	Swimming	Tennis	
Subject: Really? Swimming seems just as popular.	Swimming	Tennis	
Subject: Swimming forever It's much more fun than tennis.	Swimming	Tennis	
Subject: What?! Tennis beats swimming any day.	Swimming	Tennis	

Post-task questions

Group: _____ Participant: _____

What was your overall impression of the software you worked with?

Very negative *Very positive*

How helpful (or unhelpful) did you find the following features of the software you worked with?

Moving messages into folders

Very unhelpful *Very helpful*

Viewing number of correct predictions in each folder

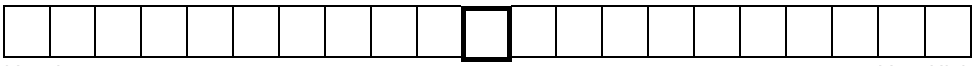
Very unhelpful *Very helpful*

Post-task questions

Group: _____ Participant: _____

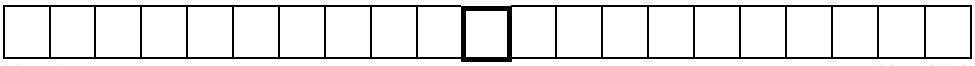
The following six questions refer to your task of trying to improve the software's topic predictions:

1. How mentally demanding was the task?



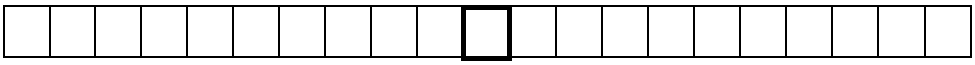
Very Low *Very High*

2. How physically demanding was the task?



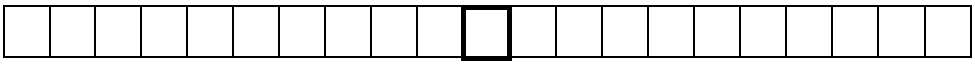
Very Low *Very High*

3. How hurried or rushed was the pace of the task?



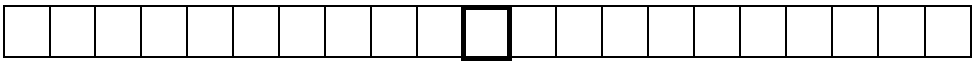
Very Low *Very High*

4. How successful were you in accomplishing what you were asked to do?



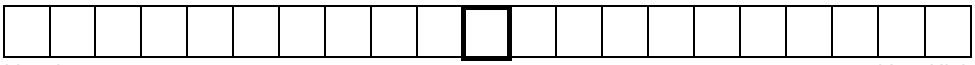
Failure *Perfect*

5. How hard did you have to work to accomplish your level of performance?



Very Low *Very High*

6. How insecure, discouraged, irritated, stressed, and annoyed were you?



Very Low *Very High*

Post-task questions

Group: _____ Participant: _____

a) Underline all of the words that you feel apply to this system.
b) Pick five underlined words that best describe your feelings toward this system and explain why.

Accessible	Creative	Fast	Meaningful	Slow
Advanced	Customizable	Flexible	Motivating	Sophisticated
Annoying	Cutting edge	Fragile	Not secure	Stable
Appealing	Dated	Fresh	Not valuable	Sterile
Approachable	Desirable	Friendly	Novel	Stimulating
Attractive	Difficult	Frustrating	Old	Straight forward
Boring	Disconnected	Fun	Optimistic	Stressful
Business-like	Disruptive	Gets in the way	Ordinary	Time-consuming
Busy	Distracting	Hard to use	Organized	Time-saving
Calm	Dull	Helpful	Overbearing	Too technical
Clean	Easy to use	High quality	Overwhelming	Trustworthy
Clear	Effective	Impersonal	Patronizing	Unapproachable
Collaborative	Efficient	Impressive	Personal	Unattractive
Comfortable	Effortless	Incomprehensible	Poor quality	Uncontrollable
Compatible	Empowering	Inconsistent	Powerful	Unconventional
Compelling	Energetic	Ineffective	Predictable	Understandable
Complex	Engaging	Innovative	Professional	Undesirable
Comprehensive	Entertaining	Inspiring	Relevant	Unpredictable
Confident	Enthusiastic	Integrated	Reliable	Unrefined
Confusing	Essential	Intimidating	Responsive	Usable
Connected	Exceptional	Intuitive	Rigid	Useful
Consistent	Exciting	Inviting	Satisfying	Valuable
Controllable	Expected	Irrelevant	Secure	
Convenient	Familiar	Low maintenance	Simplistic	

Word	Why?
1.	
2.	
3.	
4.	
5.	

Post-task questions

Group: _____ Participant: _____

If you have any additional comments or suggestions about the software you worked with today, please write them down below.

Post-task questions

Group: _____ Participant: _____

Post-task questions

Assume you are using the same software you just worked with, but instead of *Baseball* and *Hockey*, it predicts whether each message is about *Swimming* or *Tennis*. The software has learned from two messages in the *Swimming* folder, and one message in the *Tennis* folder:

Swimming messages:

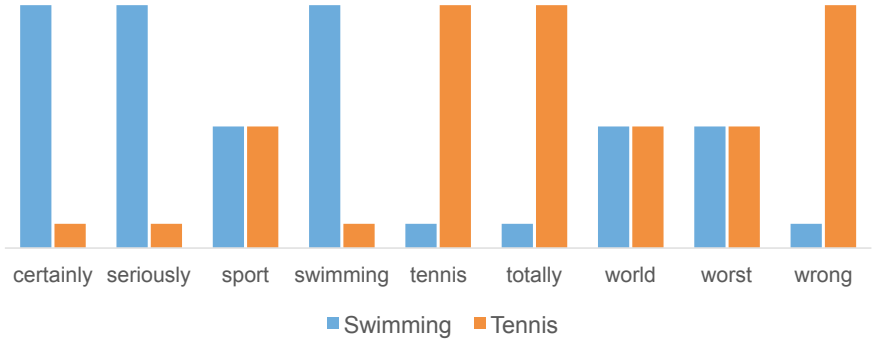
#1 **Subject: Swimming is the worst**
Swimming is certainly the worst sport in the world.

#2 **Subject: Swimming is the worst**
Seriously?

Tennis message:

#1 **Subject: Tennis is the worst**
Wrong, tennis is totally the worst sport in the world.

The **Important Words** section of the software shows the following:



Assume that **these are the only messages** the software has learned from. The following page lists several messages in the *Unknown* folder. For each message, circle the topic that you think the computer will predict and explain why. Be as specific as possible.

Group: _____ Participant: _____

Message	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules! You're wrong, rugby's awesome!	Swimming	Tennis	
Subject: Meh Chess is more exciting.	Swimming	Tennis	
Subject: Swimming rules Well it does.	Swimming	Tennis	
Subject: Yeah! Rugby totally rocks.	Swimming	Tennis	
Subject: Pshhhh But tennis is more popular.	Swimming	Tennis	
Subject: Rugby is dull Rugby is certainly the worst sport.	Swimming	Tennis	
Subject: Tennis rules Tennis is best.	Swimming	Tennis	
Subject: Chess can be exciting If you wear a helmet.	Swimming	Tennis	
Subject: Nope! Seriously, helmets do not make chess exciting.	Swimming	Tennis	
Subject: Really? Swimming seems just as popular.	Swimming	Tennis	
Subject: Swimming forever It's much more fun than tennis.	Swimming	Tennis	
Subject: What?! Tennis beats swimming any day.	Swimming	Tennis	

Post-task questions

Group: _____ Participant: _____

What was your overall impression of the software you worked with?

Very negative *Very positive*

How helpful (or unhelpful) did you find the following features of the software you worked with?

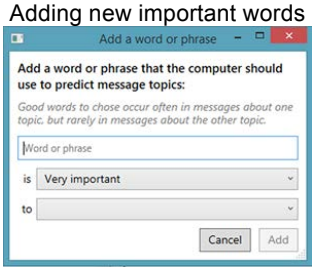
Moving messages into folders

Very unhelpful *Very helpful*

Viewing number of correct predictions in each folder

Very unhelpful *Very helpful*

Adding new important words

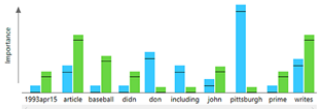


Very unhelpful *Very helpful*

Removing important worlds

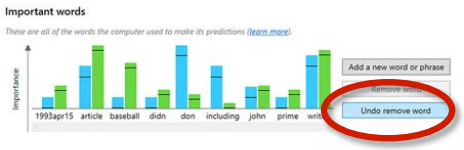
Very unhelpful *Very helpful*

Adjusting word importance



Very unhelpful *Very helpful*

Undo



Very unhelpful *Very helpful*

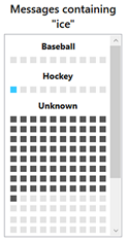
Post-task questions

Group: _____ Participant: _____

Searching for words

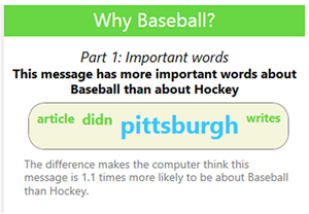
Very unhelpful *Very helpful*

Viewing which messages contain the selected word



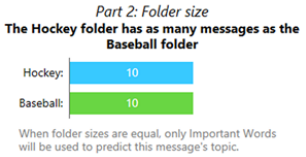
Very unhelpful *Very helpful*

Important words explanation



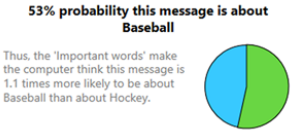
Very unhelpful *Very helpful*

Folder size explanation



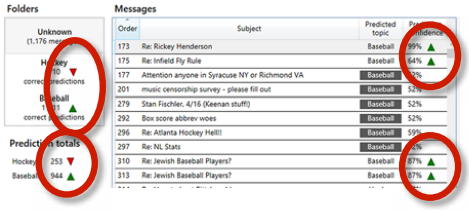
Very unhelpful *Very helpful*

Confidence explanation



Very unhelpful *Very helpful*

Arrows showing changes



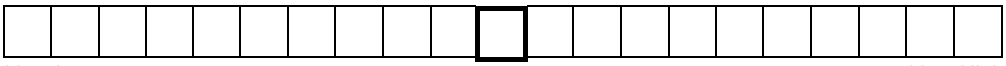
Very unhelpful *Very helpful*

Post-task questions

Group: _____ Participant: _____

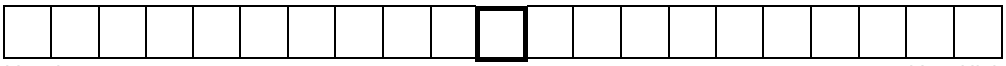
The following six questions refer to your task of trying to improve the software's topic predictions:

1. How mentally demanding was the task?



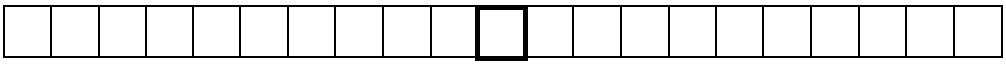
Very Low *Very High*

2. How physically demanding was the task?



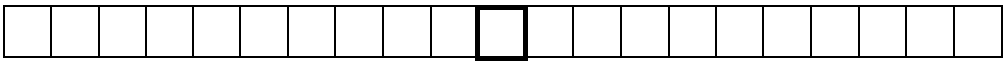
Very Low *Very High*

3. How hurried or rushed was the pace of the task?



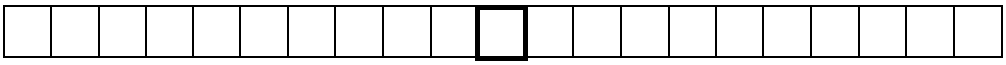
Very Low *Very High*

4. How successful were you in accomplishing what you were asked to do?



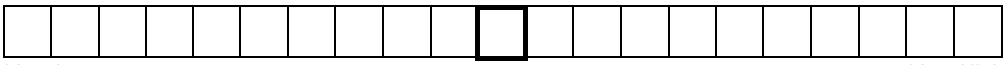
Failure *Perfect*

5. How hard did you have to work to accomplish your level of performance?



Very Low *Very High*

6. How insecure, discouraged, irritated, stressed, and annoyed were you?



Very Low *Very High*

Post-task questions

Group: _____ Participant: _____

a) Underline all of the words that you feel apply to this system.
b) Pick five underlined words that best describe your feelings toward this system and explain why.

Accessible	Creative	Fast	Meaningful	Slow
Advanced	Customizable	Flexible	Motivating	Sophisticated
Annoying	Cutting edge	Fragile	Not secure	Stable
Appealing	Dated	Fresh	Not valuable	Sterile
Approachable	Desirable	Friendly	Novel	Stimulating
Attractive	Difficult	Frustrating	Old	Straight forward
Boring	Disconnected	Fun	Optimistic	Stressful
Business-like	Disruptive	Gets in the way	Ordinary	Time-consuming
Busy	Distracting	Hard to use	Organized	Time-saving
Calm	Dull	Helpful	Overbearing	Too technical
Clean	Easy to use	High quality	Overwhelming	Trustworthy
Clear	Effective	Impersonal	Patronizing	Unapproachable
Collaborative	Efficient	Impressive	Personal	Unattractive
Comfortable	Effortless	Incomprehensible	Poor quality	Uncontrollable
Compatible	Empowering	Inconsistent	Powerful	Unconventional
Compelling	Energetic	Ineffective	Predictable	Understandable
Complex	Engaging	Innovative	Professional	Undesirable
Comprehensive	Entertaining	Inspiring	Relevant	Unpredictable
Confident	Enthusiastic	Integrated	Reliable	Unrefined
Confusing	Essential	Intimidating	Responsive	Usable
Connected	Exceptional	Intuitive	Rigid	Useful
Consistent	Exciting	Inviting	Satisfying	Valuable
Controllable	Expected	Irrelevant	Secure	
Convenient	Familiar	Low maintenance	Simplistic	

Word	Why?
1.	
2.	
3.	
4.	
5.	

Post-task questions

Group: _____ Participant: _____

If you have any additional comments or suggestions about the software you worked with today, please write them down below.

Post-task questions

Group: _____ Participant: _____

Message	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules Truly, rugby is the best sport!	Soccer	Basketball	Truly
Subject: Basketball does rule Right on!	Soccer	Basketball	Basketball
Subject: Rugby rules I think rugby is best.	Soccer	Basketball	Think
Subject: What?! Basketball beats soccer any day.	Soccer	Basketball	Class imbalance
Subject: Soccer does rule Right on!	Soccer	Basketball	Soccer
Subject: Soccer forever It's just more fun than basketball.	Soccer	Basketball	Class imbalance
Subject: Yeah! Agreed, rugby rocks.	Soccer	Basketball	Agreed
Subject: Nope! No, rugby does not rock.	Soccer	Basketball	No
Subject: Chess is better You're all wrong.	Soccer	Basketball	Class imbalance
Subject: Eh? Chess is more exciting.	Soccer	Basketball	Class imbalance
Subject: Pshhhh But soccer is more popular.	Soccer	Basketball	Soccer
Subject: Not in the states Basketball is more popular here.	Soccer	Basketball	Basketball

Group: _____ Participant: _____

Message	Computer's predicted topic		Why? (be specific)
Subject: Rugby rules! You're wrong, rugby's awesome!	Swimming	Tennis	Wrong
Subject: Meh Chess is more exciting.	Swimming	Tennis	Class imbalance
Subject: Swimming rules Well it does.	Swimming	Tennis	Swimming
Subject: Yeah! Rugby totally rocks.	Swimming	Tennis	Totally
Subject: Pshhhh But tennis is more popular.	Swimming	Tennis	Tennis
Subject: Rugby is dull Rugby is certainly the worst sport.	Swimming	Tennis	Certainly
Subject: Tennis rules Tennis is best.	Swimming	Tennis	Tennis
Subject: Chess can be exciting If you wear a helmet.	Swimming	Tennis	Class imbalance
Subject: Nope! Seriously, helmets do not make chess exciting.	Swimming	Tennis	Seriously
Subject: Really? Swimming seems just as popular.	Swimming	Tennis	Swimming
Subject: Swimming forever It's much more fun than tennis.	Swimming	Tennis	Class imbalance
Subject: What?! Tennis beats swimming any day.	Swimming	Tennis	Class imbalance