Model-Agnostic Explanations and Evaluation of Machine Learning

Marco Tulio Ribeiro

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Carlos Guestrin, Chair

Sameer Singh

Dan Weld

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Model-Agnostic Explanations and Evaluation of Machine Learning

Marco Tulio Ribeiro

Chair of the Supervisory Committee:
Ph.D Carlos Guestrin
Computer Science and Engineering

Despite many successes, complex machine learning systems are limited in their impact due to several issues regarding communication with humans: they are functionally black boxes, hard to debug and hard to evaluate properly. This communication is crucial though: humans are the ones who train, deploy and use machine learning models, and thus have to make trust and evaluation decisions. Furthermore, it is humans who try to improve these models, and having an understanding of their behavior is very valuable for this purpose. This dissertation addresses this communication problem by presenting model-agnostic explanations and evaluation, which improve the interaction between humans and *any* machine learning model.

Specifically, we present: (1) Local Interpretable Model-Agnostic Explanations (LIME), an explanation technique that can explain any black box model by approximating it locally with a linear model, (2) Anchors, model-agnostic explanations that represent sufficient conditions for predictions, (3) Semantically Equivalent Adversaries and Adversarial Rules (SEAs and SEARs), semantic-preserving perturbations and rules that unearth brittleness bugs in text models, and (4) Implication Consistency, a new kind of evaluation metric that considers the relationship between model outputs in order to measure higher level thinking. We demonstrate that these contributions enable efficient communication between machine learning models and humans, empowering humans to better evaluate, improve, and assess trust in models.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to begin by thanking my advisors Carlos Guestrin and Sameer Singh. Carlos has an uncanny ability to quickly grasp what I've done, and most importantly to notice what could be better about it. His honest and sharp criticism, coupled with his inspiring aspiration for excellence have definitely shaped how I see research. Sameer has always been amazing in various ways: by helping me both in seeing and organizing the big picture and in the nitty gritty technical details, by teaching me how to better communicate our work through writing and presenting, by teaching me how to navigate grad school, conferences, etc... I am very grateful to both Carlos and Sameer for often making me feel like an idiot - I don't think I could do any meaningful work or finish a Ph.D otherwise.

In a similar fashion, I would like to thank Isabelle Stanton, who was my manager during my first internship at Google. I had the amazing experience of working on a project for 6 weeks, only to have my spirit crushed (thanks also to Mark Paskin) at the realization that I was potentially working on the wrong problem in the first place. This was the most valuable lesson I've had in any internship, and I have always strived to think carefully about what problem I'm trying to solve and why ever since. The pains of trying to understand why my machine learning model was not working during that internship also made me interested in what is now the topic of this dissertation.

I am thankful to Renato Assunção, who first encouraged me to consider doing a Ph.D when I was quite sure I did not want to do so, and furthermore encouraged me to do it abroad, with special emphasis on UW (his alma matter).

I would also like to thank some friends in the department. My labmates Tyler Johnson and Tianqi Chen, with whom I've shared many a free lunch with great discussions around

research or politics, have taught me a lot through reading groups and discussions. Nacho Cano and Shrainik Jain helped me maintain my perspective through many coffee breaks, where we would commiserate about our lack of progress. I am glad I could always rely on Scott Lundberg to give me great feedback. He is a great friend, neighbor and an excellent discussion partner on any topic - theology, ethics, philosophy, and many others - including even research.

I certainly could not do this without the help of my faithful wife Sara. She the best companion and helper I could ever hope for, always encouraging me to be better and do things well, despite my laziness and lack of attention to aesthetics and certain details. She also helped me technically, by being an amazing proofreader and making every figure and table that looks good in any of my talks and presentations (the ones that look bad are on me).

Finally, I cannot but acknowledge the triune God, who made me and redeemed me despite my many failures. Everything in my life is meaningless apart from him, so I am thankful that he sustains me and will continue to do so to the end. To him be blessing and honor and glory forever and ever.

DEDICATION

To my parents, who have been getting me books and encouraging me to learn since before I can remember.

Chapter 1

INTRODUCTION

Machine learning (ML) is at the core of many recent advances in science and technology. ML models like neural networks have achieved equal or better than human performance in perception tasks like fine-grained image recognition [27, 87], even beating expert humans in controlled game environments like atari games [63], poker [9] and go [83]. ML models have also been successful in tasks that require higher level thinking and understanding. A task such as sentiment analysis [8] requires a model to understand nuances of human language, such as the presence of negations and sarcasm. Perhaps more challenging is the task of machine translation, hard even for trained humans, where Neural Machine Translation [5] has achieved state-of-the-art performance for various language pairs [56]. Tasks like Visual Question Answering [3, 73] and Machine Comprehension [71], where the model is given a context (an image and a paragraph, respectively) and then tasked with answering questions about it, seem to require traits like logic and reasoning, which are commonly associated with human intelligence.

As these models become more and more embedded into everyday products, and even more critically in domains like judicial decisions [80], medicine [43] and lending [39], accurate means of evaluation become crucial. Traditionally, machine learning algorithms are evaluated using a held out test set or cross-validation [57], in an attempt to avoid the problem of overfitting: having a model that performs very well on the data used for training, but does not generalize to new data. Aggregate statistics such as accuracy, precision and recall are typically employed, under the assumption that each data point is independent of the others.

Despite such important advances, or maybe in part because of them, the role of humans is an oft-overlooked aspect in the field of ML. The complexity of most state-of-the-art models

makes them functionally black-boxes, rendering the tasks of inspecting the reasoning of the model, auditing models or learning from models almost impossible. This undesirable scenario not only hinders human-ML collaboration, but may also be illegal in certain domains [23]. Even if current models were near-perfect according to standard metrics such as accuracy (which they are not), we would still have reason to be suspicious of black box models: there are many ways in which accuracy may be wrongly estimated [2, 36, 66, 81]. Furthermore, models may be prone to unacceptable behaviors such as completely changing predictions when insignificant changes are made to the input [32, 78], systematically making mistakes that go against business rules, or making predictions that are mutually inconsistent. The common thread in these (and other) problems is the need for human involvement, or in the very least human assent: humans are the ones who deploy and use machine learning models, and thus have to make trust and evaluation decisions. We now look at each of these problems in further detail, and give an overview of the solutions we propose in this dissertation, with the common goal of improving the human-ML communication and collaboration.

1.1 Problem: ML Models are Black Boxes

In contrast to sparse linear models or short decision trees (which can be visualized and grasped), the sheer model size and complexity of models such as random forests or neural networks make inspection unfeasible. The lack of communication between model and human is problematic for a variety of reasons, the most fundamental one being the issue of trust. One must trust a model explicitly by acting on its predictions (e.g. renting a movie based on a recommendation, acting on a diagnosis given by a model), or implicitly by deploying it. A user's trust is directly impacted by how much they can understand and predict the model's behavior [16], as opposed to treating it as a black box. Further, a system designer who understands why their model is making predictions is certainly better equipped to improve it by means of feature engineering, parameter tuning, or even by replacing the model with a different one (as some of our experiments will demonstrate). Understanding the behavior of the model also helps designers detect problems like feedback loops [81], data leaks [36], or

problems related to a mismatch between the metrics used for optimization and the real-world business goals (e.g. reward hacking or side effects [2]). Explanations (or rationales behind predictions) can also be a part of the model output which is useful to end users. Even in lower stakes domains such as movie or book recommendations, getting a rationale such as "you will probably like this book because of your interest in Russian Literature" makes the model much more useful to the users - a user may reject such a recommendation due to his current mood, even if overall the recommendation captures her taste. Such rationales may also be required by law in certain domains (such as lending or medicine), or for any algorithmic decision in the European Union [23].

In this dissertation, we propose two different and complementary solutions to the black box problem, both based on the concept of *explaining ML predictions*: providing a rationale for why a prediction was made using textual and visual components of the data, and/or producing counter-factual knowledge of what would have been predicted were the components different. First, we present **LIME** (short for Local Interpretable Model-Agnostic Explanations), a novel explanation technique that explains the predictions of *any* ML model in an interpretable and faithful manner, by learning an interpretable sparse linear model locally around the prediction. The linear explanations give users an easy-to-understand list of "pros and cons" for the prediction - which factors contributed to it and which factors (if any) drove the model against it. In the medical domain, for example, a machine diagnosis could be explained by what symptoms, test results and patient data are most strongly for and against the diagnosis (from the model's perspective).

Second, we present **Anchors**, high-precision explanations that represent "sufficient conditions" for model predictions. When faced with an anchor, a user knows that the presence of certain conditions in the input virtually guarantees a certain prediction, regardless of other features. In our machine diagnosis example, an example anchor would be a certain test result that is enough to warrant a model prediction, regardless of other symptoms or patient data.

We demonstrate the flexibility of these methods by explaining different models (e.g. random forests, neural networks) for different domains and tasks (e.g. text and image

classification, visual question answering, translation). We show the utility of explanations via novel experiments, both simulated and with human subjects, on various scenarios where ML-human communication is paramount: deciding if one should trust a prediction, choosing between models, improving an untrustworthy classifier, and identifying why a classifier should not be trusted. In a comparison between the two kinds of explanations, we demonstrate that they have different strengths and weaknesses, and are thus complementary.

1.2 Problem: Hard to Detect Model Brittleness

One concern when deploying ML models into the real world is whether the models will behave sensibly. While a model may seem accurate according to held-out accuracy, it may be very brittle, making different predictions for input instances that are extremely similar (a problem called oversensitivity [32]). This behavior has been observed in image classification [86], where targeted small-magnitude perturbations to images are imperceptible to the human eye, but fool models into making wrong predictions. While this poses a security risk if a malicious actor intends to fool the model for his own purposes, the issue does not compromise model performance in general, as images with this *adversarial* noise are not present apart from attacks.

Oversensitivity in text, on the other hand, is likely to happen organically. There are myriad ways of conveying the same semantic content, and a model may outputs different predictions for texts that convey the same meaning. This kind of bug is hard to detect and fix: the variability in language makes it hard to check for many possible small changes, and the cost of collecting labeled data is high even without taking this concern into account. On the other hand, being able to detect and fix this kind of behavior increases the usefulness of models in the real world, as real-world data is likely to be phrased differently than training data.

Focusing on the text domain, where humans are sensitive to how changes impact semantics, we present *semantically equivalent adversaries* (SEAs) – semantic-preserving perturbations that induce changes in models' predictions. SEAs automate part of the debugging process,

allowing a user to detect, for example, that a Visual Question Answering model gives different answers to the semantically equivalent questions "What is the person doing?" and "What's the person doing?". We generalize these adversaries into *semantically equivalent adversarial rules* (SEARs) – simple, universal replacement rules that induce adversaries on many instances. For example, the adversary presented above may be generalized into a rule that says "replace(What is, What's)". If such a rule induces many different predictions by the model, it is a good descriptor of a bug. These adversaries and bugs are easy for humans to grasp, and actionable. We demonstrate their usefulness as debugging tools in various user studies, where we show that the collaboration between human experts and SEAs and SEARs yields to much better detection of bugs than human experts alone, and that the bugs identified by SEARs can be fixed by simple data augmentation.

1.3 Problem: Evaluation does not Consider Relationships Between Outputs

Evaluation metrics for machine learning models focus on individual units in isolation, and how often the prediction for those units is correct. While important, such metrics do not capture crucial aspects of quality / intelligence as perceived by humans, for which the relationship between outputs needs to be considered. For example, one may wish to verify if members of different protected classes are treated differently by the model, all things being equal. This is not a property of individual units, but of pairs or sets of individuals. Another example typically associated with "higher level thinking" or intelligence is logical consistency - for example, a model should not predict that there are 2 people in a picture and at the same time predict that 5 of the people in the picture are males. Accuracy is rarely the only concern of a human trying to train or deploy a ML model, so being able to represent and measure human desirable properties like these is an important step in bridging the gap between human and ML model, potentially increasing trust and allowing for the discovery of improvement directions.

We present a new type of evaluation metric which uses a graph representation in order to encode relationships between different model predictions. The metric is general enough

to allow for the measurement of properties as diverse as robustness, fairness and logical consistency. As preliminary work, we focus specifically on the logical consistency of Visual Question Answering models, and propose ways of automatically expanding current datasets such that logical consistency can be evaluated. Finally, we evaluate current state of the art models for logical consistency, and show that our metric allows for the discovery of many gaps, often giving insight into directions for fixing such gaps.

1.4 Thesis Statement

We presented various challenges for human-ML interaction: the black box problem, brittleness bugs, lack of metrics to evaluate higher level thinking. Prior to our work, these problems were mostly addressed in ad-hoc or model-specific ways. Our goal in this dissertation is to address each of these problems in a model-agnostic manner - that is, not dependent on any model or type of model. This allows for our solutions to be applied to *any* model, present or future, and thus to extend the impact of our research beyond specific classes of models. We make progress towards this goal by means of explanations (loosely defined as textual or visual artifacts that provide qualitative understanding of the model) and new forms of evaluation. Our thesis is that:

Model-Agnostic explanations and evaluation enable efficient communication between machine learning models and humans, empowering humans to better evaluate, improve, and assess trust in models.

1.5 Summary of Contributions and Outline

The outline of this dissertation follows the order in which the problems were presented, with the main contributions listed below. We note that we discuss the relevant related work in each chapter, rather than in a chapter by itself.

- **[Chapter 2]**: we propose LIME, an explanation technique that can explain any black box model. Not only do we show qualitative explanations for a variety of black-box

models, we demonstrate the usefulness of LIME in multiple user studies. First, in a model selection task, non-expert users with LIME pick the model that generalizes better 89% of the time (as opposed to 68% for a baseline or 50% if choosing at random). Second, we show that non-expert users can improve a model's generalization accuracy via simple feature engineering with LIME explanations from 57% to 74% with 11 minutes of effort on average. Finally, we show that LIME explanations lead expert users to key insights, in an experiment where 44% of experts have an insight without explanations, in contrast to 93% with explanations. This chapter is based on Ribeiro et al. [74, 75].

- **[Chapter 3]**: we propose Anchors, another kind of explanation for any black box model which represents sufficient conditions for predictions. We expand the class of models for which explanations are possible to include structure prediction models (e.g. machine translation). In a user study, we show that anchors enable users to predict how a model would behave on unseen instances with over 97% precision on average, in contrast to 75% with LIME and 67% with no explanations. There is a tradeoff between LIME and anchors, as users are able to make predictions more often with LIME (68% vs 32%), but both are significantly better than no explanations, and enable users to predict what models would do much faster (on average 9 seconds for Anchor, 14 for LIME, 25 without explanations). This chapter is based on Ribeiro et al. [76].

- **[Chapter 4]**: we propose SEAs and SEARs, semantic-preserving perturbations and rules that induce changes in the model's predictions, allowing for the discovery of brittleness. We show in user studies that SEAs can generate semantically equivalent adversaries automatically as often as humans, and that users with SEARs discover brittleness bugs that impact 3 to 4 times more instances than unaided experts. We further show that we can effectively fix the bugs (73% to 88% reduction) with no discernible loss in accuracy via a simple data augmentation procedure. This chapter is based on Ribeiro et al. [77].

- **[Chapter 5]**: we propose Implication Consistency, a new kind of graph-based evaluation metric that considers the relationships between model outputs, allowing humans to indicate when certain predictions should imply other predictions. In preliminary work, we propose ways of automatically augmenting datasets for Visual Question Answering in order to check for logical consistency, and validate our augmented data by showing that humans agree with our logical implication rules on average about 98% of the time. We also display the usefulness of Implication Consistency by evaluating and displaying insights for state of the art models.

We end in **Chapter 6** by summarizing our main findings, and by outlining some of the many challenges that remain in facilitating the interaction between humans and ML models, a task that becomes more and more crucial as such models become more pervasive within our society.

Chapter 2

LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS (LIME)

2.1 Introduction

When discussing the black box problem, we noted that whether humans are directly using machine learning classifiers as tools, or are deploying models within other products, a vital concern remains: *if the users do not trust a model or a prediction, they will not use it.* It is important to differentiate between two different (but related) definitions of trust: (1) *trusting a prediction*, i.e. whether a user trusts an individual prediction sufficiently to take some action based on it, and (2) *trusting a model*, i.e. whether the user trusts a model to behave in reasonable ways if deployed. Both are directly impacted by how much the human understands a model's behaviour, as opposed to seeing it as a black box.

Determining trust in individual predictions is an important problem when the model is used for decision making. When using machine learning for medical diagnosis [11] or terrorism detection, for example, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Apart from trusting individual predictions, there is also a need to evaluate the model as a whole before deploying it "in the wild". To make this decision, users need to be confident that the model will perform well on real-world data, according to the metrics of interest. Currently, models are evaluated using accuracy metrics on an available validation dataset. However, real-world data is often significantly different, and further, the evaluation metric may not be indicative of the product's goal. Inspecting individual predictions and their explanations is a worthwhile solution, in addition to such metrics. In this case, it is important to aid users by suggesting which instances to inspect, especially for large datasets.

In this chapter, we propose providing explanations for individual predictions as a solution to the "trusting a prediction" problem, and selecting multiple such predictions (and explanations) as a solution to the "trusting the model" problem. The main contributions of this chapter are summarized as follows.

- LIME, an algorithm that can explain the predictions of *any* classifier or regressor in a faithful way, by approximating it locally with an interpretable model.
- SP-LIME, a method that selects a set of representative instances with explanations to address the "trusting the model" problem, via submodular optimization.
- Comprehensive evaluation with simulated and human subjects, where we measure the impact of explanations on trust and associated tasks. In our experiments, non-experts using LIME are able to pick which classifier from a pair generalizes better in the real world. Further, they are able to greatly improve an untrustworthy classifier trained on 20 newsgroups, by doing feature engineering using LIME. We also show how understanding the predictions of a neural network on images helps practitioners know when and why they should not trust a model.

2.2 The case for explanations

By "explaining a prediction", we mean presenting textual or visual artifacts that provide qualitative understanding of the relationship between the instance's components (e.g. words in text, patches in an image) and the model's prediction. We argue that explaining predictions is an important aspect in getting humans to trust and use machine learning effectively, if the explanations are faithful and intelligible.

The process of explaining individual predictions is illustrated in Figure 2.1. It is clear that a doctor is much better positioned to make a decision with the help of a model if intelligible explanations are provided. In this case, an explanation is a small list of symptoms with relative weights – symptoms that either contribute to the prediction (in green) or are evidence

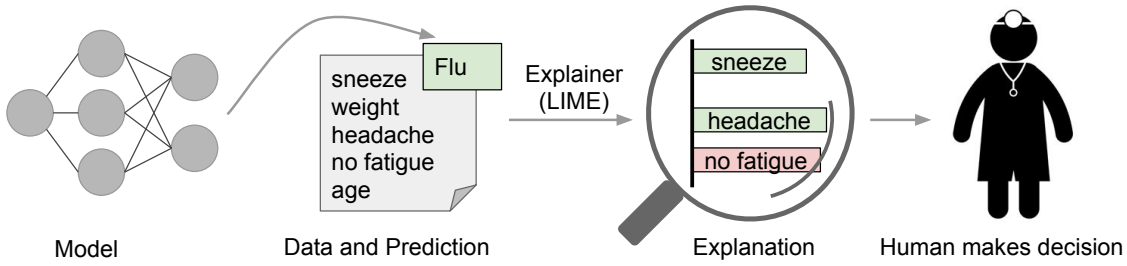


Figure 2.1: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. With these, a doctor can make an informed decision about whether to trust the model's prediction.

against it (in red). Humans usually have prior knowledge about the application domain, which they can use to accept (trust) or reject a prediction if they understand the reasoning behind it. It has been observed, for example, that providing explanations can increase the acceptance of movie recommendations [29] and other automated systems [16].

Every machine learning application also requires a certain measure of overall trust in the model. Development and evaluation of a classification model often consists of collecting annotated data, of which a held-out subset is used for automated evaluation. Although this is a useful pipeline for many applications, evaluation on validation data may not correspond to performance "in the wild", as practitioners often overestimate the accuracy of their models [66], and thus trust cannot rely solely on it. Looking at examples offers an alternative method to assess truth in the model, especially if the examples are explained. We thus propose explaining several representative individual predictions of a model as a way to provide a global understanding.

There are several ways a model or its evaluation can go wrong. Data leakage, for example, defined as the unintentional leakage of signal into the training (and validation) data that would not appear when deployed [36], potentially increases accuracy. A challenging example

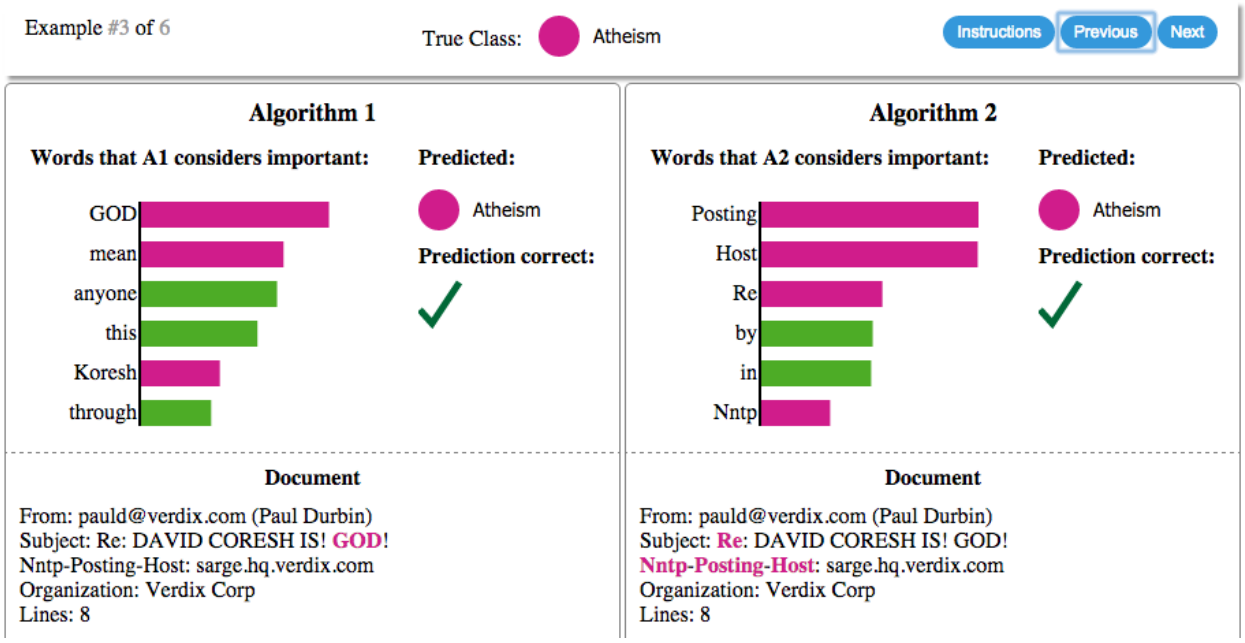


Figure 2.2: Explaining individual predictions of competing classifiers trying to determine if a document is about "Christianity" or "Atheism". The bar chart represents the importance given to the most relevant words, also highlighted in the text. Color indicates which class the word contributes to (green for "Christianity", magenta for "Atheism").

cited by Kaufman et al[36] is one where the patient ID was found to be heavily correlated with the target class in the training and validation data. This issue would be incredibly challenging to identify just by observing the predictions and the raw data, but much easier if explanations such as the one in Figure 2.1 are provided, as patient ID would be listed as an explanation for predictions. Another particularly hard to detect problem is dataset shift [10], where training data is different than test data (we give an example in the famous 20 newsgroups dataset later on). The insights given by explanations are particularly helpful in identifying what must be done to convert an untrustworthy model into a trustworthy one – for example, removing leaked data or changing the training data to avoid dataset shift.

Machine learning practitioners often have to select a model from a number of alternatives, requiring them to assess the relative trust between two or more models. In Figure 2.2, we show how individual prediction explanations can be used to select between models, in conjunction with accuracy. In this case, the algorithm with higher accuracy on the validation set is actually much worse, a fact that is easy to see when explanations are provided (again, due to human prior knowledge), but hard otherwise. Further, there is frequently a mismatch between the metrics that we can compute and optimize (e.g. accuracy) and the actual metrics of interest such as user engagement and retention. While we may not be able to measure such metrics, we have knowledge about how certain model behaviors can influence them. Therefore, a practitioner may wish to choose a less accurate model for content recommendation that does not place high importance in features related to "clickbait" articles (which may hurt user retention), even if exploiting such features increases the accuracy of the model in cross validation. We note that explanations are particularly useful in these (and other) scenarios if a method can produce them for *any* model, so that a variety of models can be compared.

Desired Characteristics for Explainers

We now outline a number of desired characteristics from explanation methods.

An essential criterion for explanations is that they must be **interpretable**, i.e., provide qualitative understanding between the input variables and the response. We note that interpretability must take into account the user's limitations. Thus, a linear model [84], a gradient vector [4] or an additive model [11] may or may not be interpretable. For example, if hundreds or thousands of features significantly contribute to a prediction, it is not reasonable to expect any user to comprehend why the prediction was made, even if individual weights can be inspected. This requirement further implies that explanations should be easy to understand, which is not necessarily true of the features used by the model, and thus the "input variables" in the explanations may need to be different than the features. Finally, we note that the notion of interpretability also depends on the target audience. Machine learning practitioners may be able to interpret small Bayesian networks, but laymen may be more

comfortable with a small number of weighted features as an explanation.

Another essential criterion is **local fidelity**. Although it is often impossible for an explanation to be completely faithful unless it is the complete description of the model itself, for an explanation to be meaningful it must at least be *locally faithful*, i.e. it must correspond to how the model behaves in the vicinity of the instance being predicted. We note that local fidelity does not imply global fidelity: features that are globally important may not be important in the local context, and vice versa. While global fidelity would imply local fidelity, identifying globally faithful explanations that are interpretable remains a challenge for complex models.

While there are models that are inherently interpretable [11, 53, 90, 92], an explainer should be able to explain *any* model, and thus be **model-agnostic** (i.e. treat the original model as a black box). We devote the next section to making a fuller argument for model-agnosticism, and note here as sufficient reason the fact that many state-of-the-art classifiers are not currently interpretable.

In addition to explaining predictions, providing a **global perspective** is important to ascertain trust in the model. As mentioned before, accuracy may often not be a suitable metric to evaluate the model, and thus we want to *explain the model*. Building upon the explanations for individual predictions, we select a few explanations to present to the user, such that they are representative of the model.

2.3 A Case for Model-Agnosticism

The prevailing solution to the black box problem is to use so called "interpretable" models, such as decision trees, rules [53, 92], additive models [11], attention-based networks [98], or sparse linear models [90]. Instead of supporting models that are functionally black-boxes, such as an arbitrary neural network or random forests with thousands of trees, these approaches use models in which there is the possibility of meaningfully inspecting model components directly — e.g. a path in a decision tree, a single rule, or the weight of a specific feature in a linear model. As long as the model is accurate for the task, and uses a reasonably

restricted number of internal components (i.e. paths, rules, or features), such approaches provide extremely useful insights.

An alternative approach to is to be *model-agnostic*, i.e. to extract post-hoc explanations by treating the original model as a black box. This involves learning an interpretable model on the predictions of the black box model [4, 13], perturbing inputs and seeing how the black box model reacts [47, 84], or both. It is for this approach that we argue for in this section, noting that restricting the space of models to be interpretable is a constraint that results in less flexibility, accuracy, and usability.

2.3.1 Model Flexibility

For most real-world applications, it is necessary to train models that are accurate for the task, irrespective of how complex or uninterpretable the underlying mechanism may be. We can observe this ideology manifesting with the increasing commonplace deployment of uninterpretable deep neural architectures for a wide variety of tasks.

Interpretable models for such tasks remain unsatisfying; such models are inherently crippled by the need to be understandable, being susceptible to the limited "perception budget" [60] of the users. This trade-off between model flexibility and interpretability [20] implies one cannot use a model whose behavior is very complex, yet expect humans to fully comprehend it globally. For example, for a task such as predicting the sentiment of a sentence, producing an accurate model that is understandable seems like an unfeasible task. The size of the vocabulary alone makes it impossible for a short set of rules, a decision tree, or an additive model to be sufficiently accurate, not to mention more complex word interactions such as negation. Tasks that involve sensory data, such as audio and images, also suffer from the same problem: for a model to be useful, it must be sufficiently flexible to handle the data complexity.

In model-agnostic interpretability, the model is treated as a black box. The separation of interpretability from the model thus frees up the model to be as flexible as necessary for the task, enabling the use of any machine learning approach - including, for example, arbitrary

deep neural networks. It also allows for the control of the complexity-interpretability trade-off (see next section), or "failing gracefully" if an interpretable explanation is not possible.

2.3.2 Explanation Flexibility

Different kinds of explanations meet different information needs. In some cases, users may only care about positive evidence towards a certain prediction (e.g. which part of an image is most responsible for the prediction), while in other instances knowing the negative evidence may be useful (e.g. in debugging a classifier). Yet in other cases, the information need may be of counter-factuals, e.g. how the model would behave if certain features had different values. Different users may also be able to handle different kinds of explanations; a user trained in statistics may be able to understand a Bayesian network, while a linear model is more intuitive to the layman. Even if the explanation type is kept fixed, users may tolerate different granularities in different situations. For example, Freitas [20] notes a case where 41 rules are considered overwhelming, and contrasts it to another user who patiently analyzed 29,050 rules.

Most interpretable models are, however, restricted in what explanations are possible, be it a prototype [40], a set of rules [53] or line graphs [11]. Further, other constraints on interpretability, such as granularity, also have to be set *a priori* (e.g. max number of rules). On the other hand, by keeping the model separate from the explanations, one is able to tailor the explanation to the information need, while keeping the model fixed. If it is possible to measure how faithful the explanation is to the original model, one can effectively control the trade-off between fidelity and interpretability, as favored by Freitas [20]. Such approaches may also be able to provide multiple explanations of different types to the user, perhaps automatically picking the one with the highest faithfulness. Thus, by being model-agnostic, the same model can be explained with different types of explanations, and different degrees of interpretability for each type of explanation. As an example, we propose two different types of explanations that are complementary in this chapter and in Chapter 3, both of which can be used to explain the same model.

2.3.3 Representation Flexibility

In domains such as images, audio and text, many of the features used to represent instances in state-of-the-art solutions are themselves not interpretable. Unsupervised feature learning produces representations such as word embeddings [59], or the so-called deep features [104]. While an interpretable model trained on such features is still uninterpretable, model-agnostic approaches can generate explanations using different features than the one used by the underlying model. Thus, even if the model is using word embeddings, the explanations can be in terms of words, for example.

2.3.4 Lower Cost to Switch

Switching models is not an uncommon operation in machine learning pipelines. If one commits to using an interpretable model, one is "locked-in" to a particular model and a particular kind of explanations - even if newer, more accurate models are developed. Even when the switch is from one interpretable model to another, users may have to be re-trained in understanding the new explanations, and the model's utility may decrease due to cognitive overhead. In contrast, if one uses model-agnostic explanations, switching the underlying model for a new one is trivial, while the way in which the explanations are presented is maintained.

2.3.5 Comparing Two Models

When deploying machine learning in the real world, a system designer often has to decide between one or more contenders, and an incumbent model. This comparison is hard to do if any of the systems are using interpretable models, while others are not. Further, even if all of the models are interpretable, it may still be difficult to compare the insights gained from each if the underlying explanations are different in their representation - for example comparing a rule-based model with a tree-based model. It is also not clear what to do if one of the contenders is less accurate but more interpretable, or vice versa. With model-agnostic explanations, the models being compared can be explained using the same techniques and

representations, as we in fact do in Section 2.7.

2.3.6 Challenges

While we have made a case for model-agnosticism, this approach is not without its challenges. Model-agnosticism necessarily requires getting multiple predictions from the model in order to discover its boundaries and behavior, which comes at an added computational cost. In some domains, exact explanations may be required (e.g. for legal or ethical reasons), and using a black-box may be unacceptable (or even illegal). Interpretable models may also be more desirable when interpretability is much more important than accuracy, or when interpretable models trained on a small number of carefully engineered features are as accurate as black-box models.

Despite the challenge of generating local explanations (which we have not addressed yet), getting a global understanding of the model may be hard if the model is very complex, due to the trade-off between flexibility and interpretability. To make matters worse, local explanations may seem inconsistent with one another, since a flexible model may use a certain feature in different ways depending on the other features. We address both of these issues in the next sections and in Chapter 3.

2.4 Local Interpretable Model-Agnostic Explanations

We now present Local Interpretable Model-agnostic Explanations (**LIME**). The overall goal of LIME is to identify an **interpretable** model over the *interpretable representation* that is **locally faithful** to the classifier.

2.4.1 Interpretable Data Representations

Before we present the explanation system, it is important to distinguish between features and interpretable data representations. As mentioned before, **interpretable** explanations need to use a representation that is understandable to humans, regardless of the actual features used by the model. For example, a possible *interpretable representation* for text classification

is a binary vector indicating the presence or absence of a word, even though the classifier may use more complex (and incomprehensible) features such as word embeddings. Likewise for image classification, an *interpretable representation* may be a binary vector indicating the "presence" or "absence" of a contiguous patch of similar pixels (a super-pixel), while the classifier may represent the image as a tensor with three color channels per pixel. We denote $x \in \mathbb{R}^d$ be the original representation of an instance being explained, and we use $x' \in \{0,1\}^{d'}$ to denote a binary vector for its interpretable representation.

2.4.2 Fidelity-Interpretability Trade-off

Formally, we define an explanation as a model $g \in G$, where G is a class of potentially *interpretable* models, such as linear models, decision trees, or falling rule lists [92], i.e. a model $g \in G$ can be readily presented to the user with visual or textual artifacts. The domain of g is $\{0,1\}^{d'}$, i.e. g acts over absence/presence of the *interpretable components*. As not every $g \in G$ may be simple enough to be interpretable - thus we let $\Omega(g)$ be a measure of *complexity* (as opposed to *interpretability*) of the explanation $g \in G$. For example, for decision trees $\Omega(g)$ may be the depth of the tree, while for linear models, $\Omega(g)$ may be the number of non-zero weights.

Let the model being explained be denoted $f : \mathbb{R}^d \to \mathbb{R}$. In classification, $f(x)$ is the probability (or a binary indicator) that x belongs to a certain class[1]. We further use $\pi_x(z)$ as a proximity measure between an instance z to x, so as to define locality around x. Finally, let $\mathcal{L}(f, g, \pi_x)$ be a measure of how unfaithful g is in approximating f in the locality defined by π_x. In order to ensure both **interpretability** and **local fidelity**, we must minimize $\mathcal{L}(f, g, \pi_x)$ while having $\Omega(g)$ be low enough to be interpretable by humans. The explanation produced by **LIME** is obtained by the following:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \quad \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.1}$$

This formulation can be used with different explanation families G, fidelity functions $\mathcal{L}$, and

[1]For multiple classes, we explain each class separately, thus $f(x)$ is the prediction of the relevant class.

complexity measures Ω. Here we focus on sparse linear models as explanations, and on performing the search using perturbations.

2.4.3 Sampling for Local Exploration

We want to minimize the locality-aware loss $\mathcal{L}(f, g, \pi_x)$ without making any assumptions about f, since we want the explainer to be **model-agnostic**. Thus, in order to learn the local behavior of f as the interpretable inputs vary, we approximate $\mathcal{L}(f, g, \pi_x)$ by drawing samples, weighted by π_x. We sample instances around x' by drawing nonzero elements of x' uniformly at random (where the number of such draws is also uniformly sampled). Given a perturbed sample $z' \in \{0,1\}^{d'}$ (which contains a fraction of the nonzero elements of x'), we recover the sample in the original representation $z \in R^d$ and obtain $f(z)$, which is used as a *label* for the explanation model. Given this dataset $\mathcal{Z}$ of perturbed samples with the associated labels, we optimize Eq. (2.1) to get an explanation $\xi(x)$. The primary intuition behind LIME is presented in Figure 2.3, where we sample instances both in the vicinity of x (which have a high weight due to π_x) and far away from x (low weight from π_x). Even though the original model may be too complex to explain globally, LIME presents an explanation that is locally faithful (linear in this case), where the locality is captured by π_x. It is worth noting that our method is fairly robust to sampling noise since the samples are weighted by π_x in Eq. (2.1). We now present a concrete instance of this general framework.

2.4.4 Sparse Linear Explanations

For the rest of this chapter, we let G be the class of linear models, such that $g(z') = w_g \cdot z'$. We use the locally weighted square loss as $\mathcal{L}$, as defined in Eq. (2.2), where we let $\pi_x(z) = exp(-D(x,z)^2/\sigma^2)$ be an exponential kernel defined on some distance function D (e.g. cosine distance for text, $L2$ distance for images) with width σ.

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2 \tag{2.2}$$

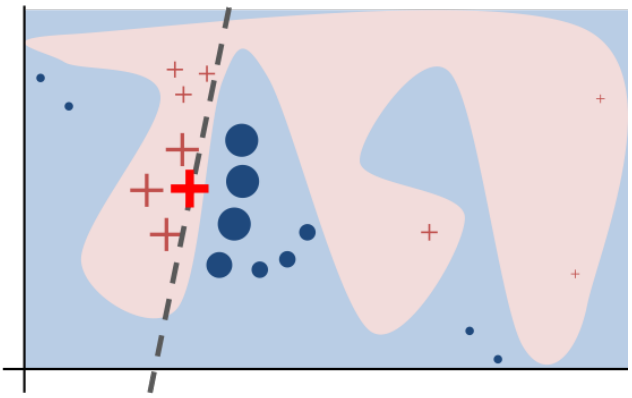


Figure 2.3: Toy example to present intuition for LIME. The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained. LIME samples instances, gets predictions using f, and weighs them by the proximity to the instance being explained (represented here by size). The dashed line is the learned explanation that is locally (but not globally) faithful.

For text classification, we ensure that the explanation is **interpretable** by letting the *interpretable representation* be a bag of words, and by setting a limit K on the number of words, i.e. $\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > K]$. Potentially, K can be adapted to be as big as the user can handle, or we could have different values of K for different instances. In this chapter we use a constant value for K, leaving the exploration of different values to future work. We use the same Ω for image classification, using "super-pixels" (computed using any standard algorithm) instead of words, such that the interpretable representation of an image is a binary vector where 1 indicates the original super-pixel and 0 indicates a grayed out super-pixel. This particular choice of Ω makes directly solving Eq. (2.1) intractable, but we approximate it by first selecting K features with Lasso (using the regularization path [18]) and then learning the weights via least squares (a procedure we call K-LASSO in Algorithm 1). Since Algorithm 1 produces an explanation for an individual prediction, its complexity does not depend on the size of the dataset, but instead on time to compute $f(x)$ and on

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N

Require: Instance x, and its interpretable version x'

Require: Similarity kernel π_x, Length of explanation K

$\mathcal{Z} \leftarrow \{\}$

for $i \in \{1, 2, 3, ..., N\}$ **do**

$z'_i \leftarrow sample\_around(x')$

$\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$

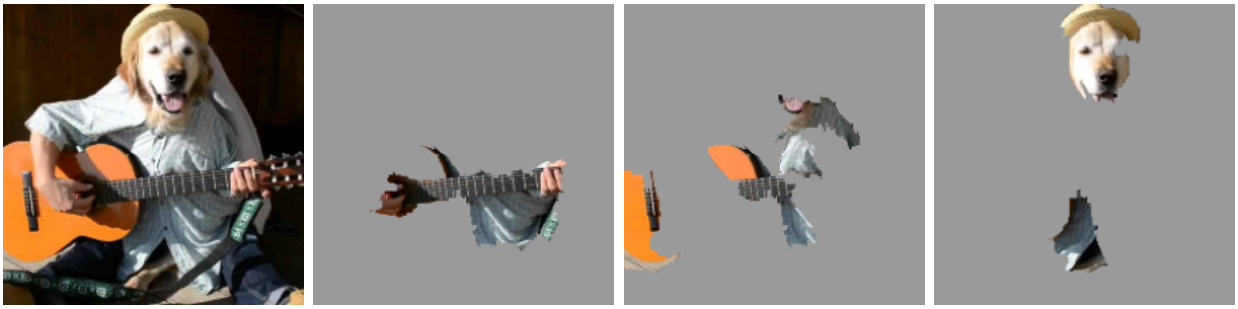
end for

$w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ ▷ with z'_i as features, $f(z)$ as target

return w

the number of samples N. In practice, explaining random forests with 1000 trees using scikit-learn (`http://scikit-learn.org`) on a laptop with $N = 5000$ takes under 3 seconds without any optimizations such as using gpus or parallelization. Explaining each prediction of the Inception network [87] for image classification takes around 10 minutes.

Any choice of interpretable representations and G will have some inherent drawbacks. First, while the underlying model can be treated as a black-box, certain interpretable representations will not be powerful enough to explain certain behaviors. For example, a model that predicts sepia-toned images to be *retro* cannot be explained by presence of absence of super pixels. Second, our choice of G (sparse linear models) means that if the underlying model is highly non-linear even in the locality of the prediction, there may not be a faithful explanation. However, we can estimate the faithfulness of the explanation on $\mathcal{Z}$, and present this information to the user. This estimate of faithfulness can also be used for selecting an appropriate family of explanations from a set of multiple interpretable model classes, thus adapting to the given dataset and the classifier. In this chapter, we restrict ourselves to linear explanations, which work quite well for multiple black-box models in our experiments.



(a) **Original Image** (b) ***Electric guitar*** (c) ***Acoustic guitar*** (d) ***Labrador***

Figure 2.4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$)**, "Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)**.**

2.4.5 Example 1: Text classification with SVMs

In Figure 2.2 (right side), we explain the predictions of a support vector machine with RBF kernel trained on unigrams to differentiate "Christianity" from "Atheism" (on a subset of the 20 newsgroup dataset). Although this classifier achieves 94% held-out accuracy, and one would be tempted to trust it based on this, the explanation for an instance shows that predictions are made for quite arbitrary reasons (words "Posting", "Host", and "Re" have no connection to either Christianity or Atheism). The word "Posting" appears in 22% of examples in the training set, 99% of them in the class "Atheism". Even if headers are removed, proper names of prolific posters in the original newsgroups are selected by the classifier, which would also not generalize.

After getting such insights from explanations, it is clear that this dataset has serious issues (which are not evident just by studying the raw data or predictions), and that this classifier, or held-out evaluation, cannot be trusted. It is also clear what the problems are, and the steps that can be taken to fix these issues and train a more trustworthy classifier.

2.4.6 Example 2: Deep networks for images

When using sparse linear explanations for image classifiers, one may wish to just highlight the super-pixels with positive weight towards a specific class, as they give intuition as to why the model would think that class may be present. We explain the prediction of Google's pre-trained Inception neural network [87] in this fashion on an arbitrary image (Figure 2.4a). Figures 2.4b, 2.4c, 2.4d show the superpixels explanations for the top 3 predicted classes (with the rest of the image grayed out), having set $K = 10$. What the neural network picks up on for each of the classes is quite natural to humans - Figure 2.4b in particular provides insight as to why acoustic guitar was predicted to be electric: due to the fretboard. This kind of explanation enhances trust in the classifier (even if the top predicted class is wrong), as it shows that it is not acting in an unreasonable manner.

2.5 Submodular Pick for Explaining Models

Although an explanation of a single prediction provides some understanding into the reliability of the classifier to the user, it is not sufficient to evaluate and assess trust in the model as a whole. We propose to give a global understanding of the model by explaining a set of individual instances. This approach is still model-agnostic, and is complementary to computing summary statistics such as held-out accuracy.

Even though explanations of multiple instances can be insightful, these instances need to be selected judiciously, since users may not have the time to examine a large number of explanations. We represent the time/patience that humans have by a budget B that denotes the number of explanations they are willing to look at in order to understand a model. Given a set of instances X, we define the **pick step** as the task of selecting B instances for the user to inspect.

The pick step is not dependent on the existence of explanations - one of the main purpose of tools like Modeltracker [1] and others [25] is to assist users in selecting instances themselves, and examining the raw data and predictions. However, since looking at raw data is not enough to understand predictions and get insights, the pick step should take into account the

explanations that accompany each prediction. Moreover, this method should pick a diverse, representative set of explanations to show the user – i.e. non-redundant explanations that represent how the model behaves globally.

Given the explanations for a set of instances X ($|X| = n$), we construct an $n \times d'$ *explanation matrix* $\mathcal{W}$ that represents the local importance of the interpretable components for each instance. When using linear models as explanations, for an instance x_i and explanation $g_i = \xi(x_i)$, we set $\mathcal{W}_{ij} = |w_{g_{ij}}|$. Further, for each component (column) j in $\mathcal{W}$, we let I_j denote the *global* importance of that component in the explanation space. Intuitively, we want I such that features that explain many different instances have higher importance scores. In Figure 2.5, we show a toy example $\mathcal{W}$, with $n = d' = 5$, where $\mathcal{W}$ is binary (for simplicity). The importance function I should score feature f2 higher than feature f1, i.e. $I_2 > I_1$, since feature f2 is used to explain more instances. Concretely for the text applications, we set $I_j = \sqrt{\sum_{i=1}^{n} \mathcal{W}_{ij}}$. For images, I must measure something that is comparable across the super-pixels in different images, such as color histograms or other features of super-pixels; we leave further exploration of these ideas for future work.

While we want to pick instances that cover the important components, the set of explanations must not be redundant in the components they show the users, i.e. avoid selecting instances with similar explanations. In Figure 2.5, after the second row is picked, the third row adds no value, as the user has already seen features f2 and f3 - while the last row exposes the user to completely new features. Selecting the second and last row results in the coverage of almost all the features. We formalize this non-redundant coverage intuition in Eq. (2.3), where we define coverage as the set function c that, given $\mathcal{W}$ and I, computes the total importance of the features that appear in at least one instance in a set V.

$$c(V, \mathcal{W}, I) = \sum_{j=1}^{d'} I_j \mathbb{1}\big[\exists i \in V : \mathcal{W}_{ij} > 0\big] \tag{2.3}$$

The pick problem, defined in Eq. (2.4), consists of finding the set $V, |V| \leq B$ that achieves

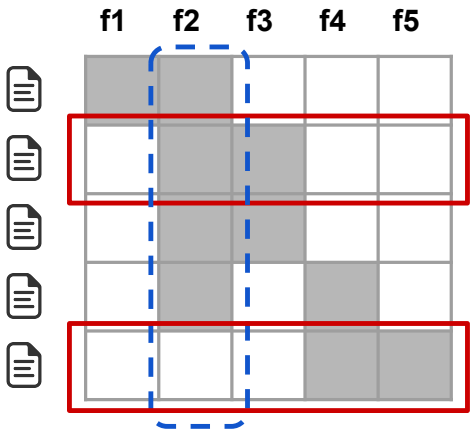


Figure 2.5: Toy example $\mathcal{W}$. Rows represent instances (documents) and columns represent features (words). Feature f2 (dotted blue) has the highest importance. Rows 2 and 5 (in red) would be selected by the pick procedure, covering all but feature f1.

highest coverage.

$$Pick(\mathcal{W}, I) = \underset{V, |V| \leq B}{\operatorname{argmax}} c(V, \mathcal{W}, I) \tag{2.4}$$

The problem in Eq. (2.4) is maximizing a weighted coverage function, and is NP-hard [19]. Let $c(V \cup \{i\}, \mathcal{W}, I) - c(V, \mathcal{W}, I)$ be the marginal coverage gain of adding an instance i to a set V. Due to submodularity, a greedy algorithm that iteratively adds the instance with the highest marginal coverage gain to the solution offers a constant-factor approximation guarantee of $1 - 1/e$ to the optimum [45]. We outline this approximation in Algorithm 2, and call it **submodular pick**.

2.6 Simulated User Experiments

In this section, we present simulated user experiments to evaluate the utility of explanations in trust-related tasks. In particular, we address the following questions: (1) Are the explanations faithful to the model, (2) Can the explanations aid users in ascertaining trust in predictions,

Algorithm 2 Submodular pick (SP) algorithm

Require: Instances X, Budget B

for all $x_i \in X$ **do**

$\mathcal{W}_i \leftarrow$ **explain**(x_i, x_i') ▷ Using Algorithm 1

end for

for $j \in \{1 \dots d'\}$ **do**

$I_j \leftarrow \sqrt{\sum_{i=1}^{n} |\mathcal{W}_{ij}|}$ ▷ Compute feature importances

end for

$V \leftarrow \{\}$

while $|V| < B$ **do** ▷ Greedy optimization of Eq (2.4)

$V \leftarrow V \cup \text{argmax}_i \, c(V \cup \{i\}, \mathcal{W}, I)$

end while

return V

and (3) Are the explanations useful for evaluating the model as a whole.

2.6.1 Experiment Setup

We use two sentiment analysis datasets (*books* and *DVDs*, 2000 instances each) where the task is to classify product reviews as positive or negative [8]. We train decision trees (**DT**), logistic regression with L2 regularization (**LR**), nearest neighbors (**NN**), and support vector machines with RBF kernel (**SVM**), all using bag of words as features. We also include random forests (with 1000 trees) trained with the average word2vec embedding [59] (**RF**), a model that is impossible to interpret without a technique like LIME. We use the implementations and default parameters of scikit-learn, unless noted otherwise. We divide each dataset into train (1600 instances) and test (400 instances).

To explain individual predictions, we compare our proposed approach (**LIME**), with **parzen** [4], a method that approximates the black box classifier globally with Parzen windows, and explains individual predictions by taking the gradient of the prediction probability function.

For parzen, we take the K features with the highest absolute gradients as explanations. We set the hyper-parameters for parzen and LIME using cross validation, and set $N = 15,000$. We also compare against a **greedy** procedure (similar to Martens and Provost [58]) in which we greedily remove features that contribute the most to the predicted class until the prediction changes (or we reach the maximum of K features), and a **random** procedure that randomly picks K features as an explanation. We set K to 10 for our experiments.

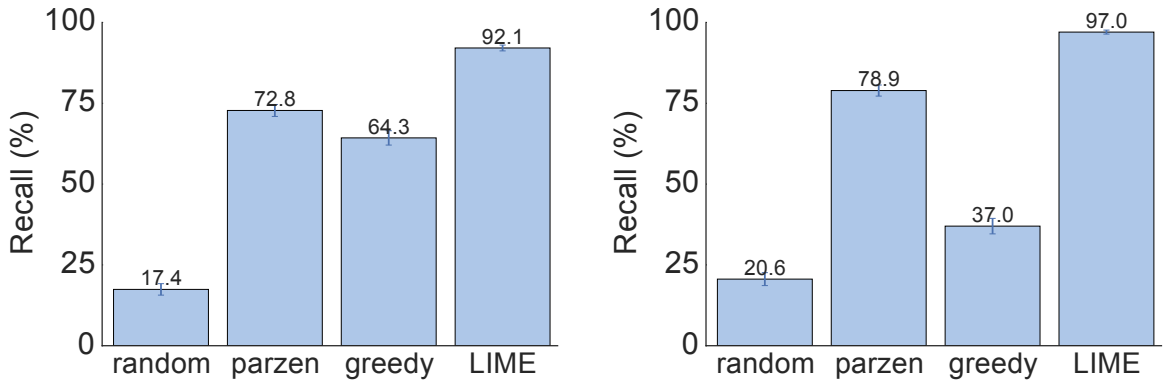
For experiments where the pick procedure applies, we either do random selection (random pick, **RP**) or the procedure described in §2.5 (submodular pick, **SP**). We refer to pick-explainer combinations by adding RP or SP as a prefix.

2.6.2 Are explanations faithful to the model?

We measure faithfulness of explanations on classifiers that are by themselves interpretable (sparse logistic regression and decision trees). In particular, we train both classifiers such that the maximum number of features they use for any instance is 10, and thus we know the *gold* set of features that the are considered important by these models. For each prediction on the test set, we generate explanations and compute the fraction of these *gold* features that are recovered by the explanations. We report this recall averaged over all the test instances in Figures 2.6 and 2.7. We observe that the greedy approach is comparable to parzen on logistic regression, but is substantially worse on decision trees since changing a single feature at a time often does not have an effect on the prediction. The overall recall by parzen is low, likely due to the difficulty in approximating the original high-dimensional classifier. LIME consistently provides $> 90\%$ recall for both classifiers on both datasets, demonstrating that LIME explanations are faithful to the models.

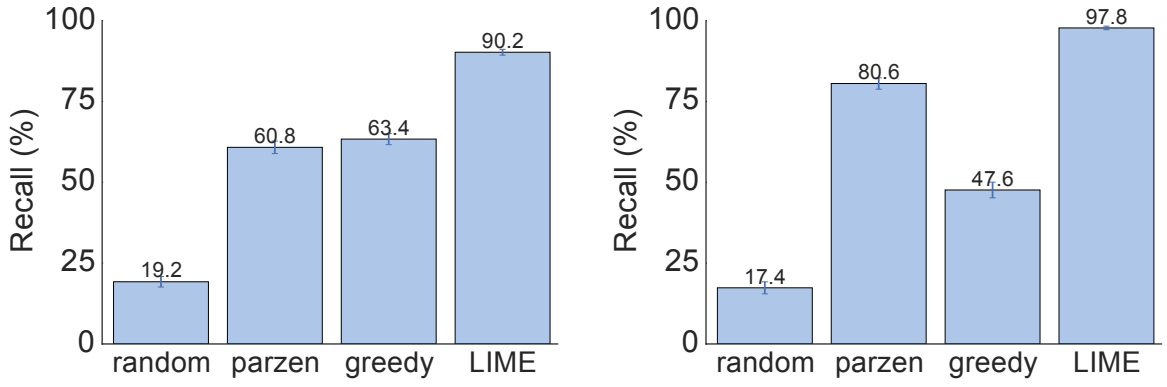
2.6.3 Should I trust this prediction?

In order to simulate trust in individual predictions, we first randomly select 25% of the features to be "untrustworthy", and assume that the users can identify and would not want to trust these features (such as the headers in 20 newsgroups, leaked data, etc). We thus



(a) Sparse LR **(b) Decision Tree**

Figure 2.6: Recall on truly important features for two interpretable classifiers on the books dataset.



(a) Sparse LR **(b) Decision Tree**

Figure 2.7: Recall on truly important features for two interpretable classifiers on the DVDs dataset.

develop *oracle* "trustworthiness" by labeling test set predictions from a black box classifier as "untrustworthy" if the prediction changes when untrustworthy features are removed from the instance, and "trustworthy" otherwise. In order to simulate users, we assume that users deem predictions untrustworthy from LIME and parzen explanations if the prediction from the linear approximation changes when all untrustworthy features that appear in the explanations are removed (the simulated human "discounts" the effect of untrustworthy features). For greedy and random, the prediction is mistrusted if any untrustworthy features are present in the explanation, since these methods do not provide a notion of the contribution of each feature to the prediction. Thus for each test set prediction, we can evaluate whether the simulated user trusts it using each explanation method, and compare it to the trustworthiness oracle.

Using this setup, we report the F1 on the trustworthy predictions for each explanation method, averaged over 100 runs, in Table 2.1. The results indicate that LIME dominates others (all results are significant at $p = 0.01$) on both datasets, and for all of the black box models. The other methods either achieve a lower recall (i.e. they mistrust predictions more than they should) or lower precision (i.e. they trust too many predictions), while LIME maintains both high precision and high recall. Even though we artificially select which features are untrustworthy, these results indicate that LIME is helpful in assessing trust in individual predictions.

2.6.4 Can I trust this model?

In the final simulated user experiment, we evaluate whether the explanations can be used for model selection, simulating the case where a human has to decide between two competing models with similar accuracy on validation data. For this purpose, we add 10 artificially "noisy" features. Specifically, on training and validation sets (80/20 split of the original training data), each artificial feature appears in 10% of the examples in one class, and 20% of the other, while on the test instances, each artificial feature appears in 10% of the examples in each class. This recreates the situation where the models use not only features that are

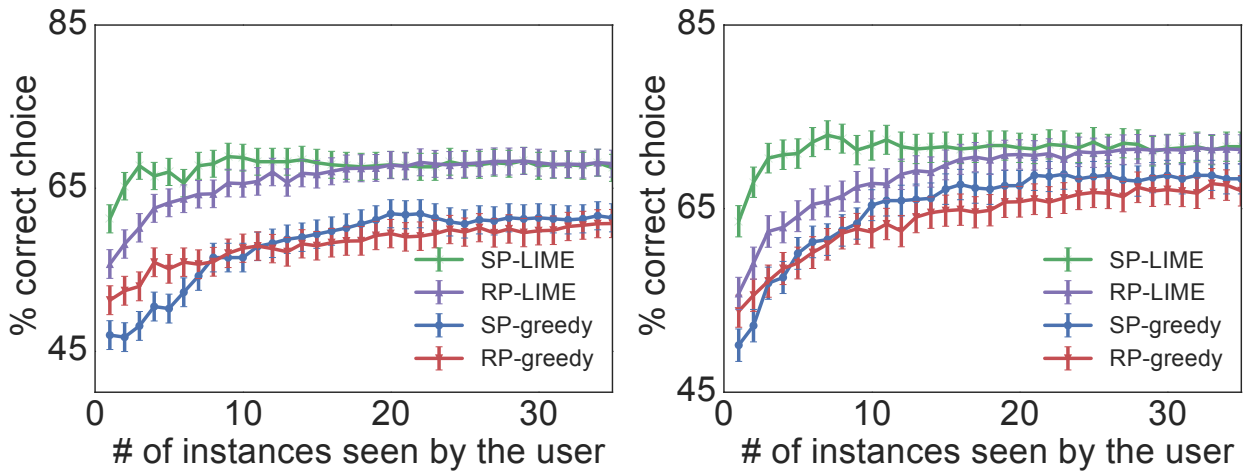
Table 2.1: Average F1 of *trustworthiness* for different explainers on a collection of classifiers and datasets.

	Books				**DVDs**			
	LR	NN	RF	SVM	LR	NN	RF	SVM
Random	14.6	14.8	14.7	14.7	14.2	14.3	14.5	14.4
Parzen	84.0	87.6	94.3	92.3	87.0	81.7	94.2	87.3
Greedy	53.7	47.4	45.0	53.3	52.4	58.1	46.6	55.1
LIME	**96.6**	**94.5**	**96.2**	**96.7**	**96.6**	**91.8**	**96.1**	**95.6**

informative in the real world, but also ones that introduce spurious correlations. We create pairs of competing classifiers by repeatedly training pairs of random forests with 30 trees until their validation accuracy is within 0.1% of each other, but their test accuracy differs by at least 5%. Thus, it is not possible to identify the *better* classifier (the one with higher test accuracy) from the accuracy on the validation data.

The goal of this experiment is to evaluate whether a user can identify the better classifier based on the explanations of B instances from the validation set. The simulated human marks the set of artificial features that appear in the B explanations as untrustworthy, following which we evaluate how many total predictions in the validation set should be trusted (as in the previous section, treating only marked features as untrustworthy). Then, we select the classifier with fewer untrustworthy predictions, and compare this choice to the classifier with higher held-out test set accuracy.

We present the accuracy of picking the correct classifier as B varies, averaged over 800 runs, in Figure 2.8. We omit SP-parzen and RP-parzen from the figure since they did not produce useful explanations, performing only slightly better than random. LIME is consistently better than greedy, irrespective of the pick method. Further, combining submodular pick with LIME



(a) Books dataset **(b) DVDs dataset**

Figure 2.8: Choosing between two classifiers, as the number of instances shown to a simulated user is varied. Averages and standard errors from 800 runs.

outperforms all other methods, in particular it is much better than RP-LIME when only a few examples are shown to the users. These results demonstrate that the trust assessments provided by SP-selected LIME explanations are good indicators of generalization, which we validate with human experiments in the next section.

2.7 Evaluation with human subjects

In this section, we recreate three scenarios in machine learning that require trust and understanding of predictions and models. In particular, we evaluate LIME and SP-LIME in the following settings: (1) Can users choose which of two classifiers generalizes better (Section 2.7.2), (2) based on the explanations, can users perform feature engineering to improve the model (Section 2.7.3), and (3) are users able to identify and describe classifier irregularities by looking at explanations (Section 2.7.4).

2.7.1 Experiment setup

For experiments in in Sections 2.7.2 and 2.7.3, we use the "Christianity" and "Atheism" documents from the 20 newsgroups dataset mentioned beforehand. This dataset is problematic since it contains features that do not generalize (e.g. very informative header information and author names), and thus validation accuracy considerably overestimates real-world performance.

In order to estimate the real world performance, we create a new *religion dataset* for evaluation. We download Atheism and Christianity websites from the DMOZ directory and human curated lists, yielding 819 webpages in each class. High accuracy on this dataset by a classifier trained on 20 newsgroups indicates that the classifier is generalizing using semantic content, instead of placing importance on the data specific issues outlined above. Unless noted otherwise, we use SVM with RBF kernel, trained on the 20 newsgroups data with hyper-parameters tuned via the cross-validation.

2.7.2 Can users select the best classifier?

In this section, we want to evaluate whether explanations can help users decide which classifier generalizes better, i.e., which classifier would the user deploy "in the wild". Specifically, users have to decide between two classifiers: SVM trained on the original 20 newsgroups dataset, and a version of the same classifier trained on a "cleaned" dataset where many of the features that do not generalize have been manually removed. The original classifier achieves an accuracy score of 57.3% on the *religion dataset*, while the "cleaned" classifier achieves a score of 69.0%. In contrast, the test accuracy on the original 20 newsgroups split is 94.0% and 88.6%, respectively – suggesting that the worse classifier would be selected if accuracy alone is used as a measure of trust.

We recruit human subjects on Amazon Mechanical Turk – by no means machine learning experts, but instead people with basic knowledge about religion. We measure their ability to choose the better algorithm by seeing side-by-side explanations with the associated raw data

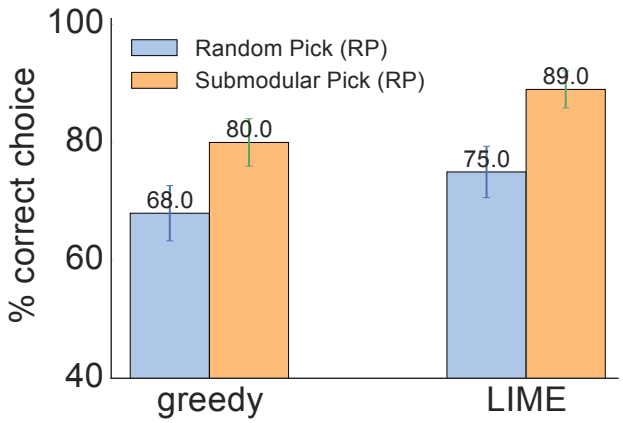


Figure 2.9: Average accuracy of human subject (with standard errors) in choosing between two classifiers.

(as shown in Figure 2.2). We restrict both the number of words in each explanation (K) and the number of documents that each person inspects (B) to 6. The position of each algorithm and the order of the instances seen are randomized between subjects. After examining the explanations, users are asked to select which algorithm will perform best in the real world. The explanations are produced by either greedy (chosen as a baseline due to its performance in the simulated user experiment) or LIME, and the instances are selected either by random (RP) or submodular pick (SP). We modify the greedy step in Algorithm 2 slightly so it alternates between explanations of the two classifiers. For each setting, we repeat the experiment with 100 users.

The results are presented in Figure 2.9. Note that all of the methods are good at identifying the better classifier, demonstrating that the explanations are useful in determining which classifier to trust, while using test set accuracy would result in the selection of the wrong classifier. Further, we see that the submodular pick (SP) greatly improves the user's ability to select the best classifier when compared to random pick (RP), with LIME outperforming greedy in both cases.

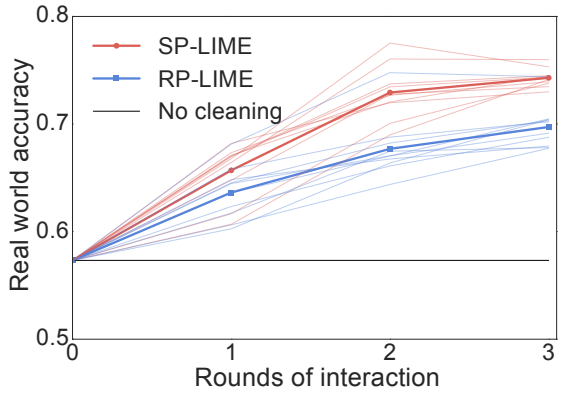


Figure 2.10: Feature engineering experiment. Each shaded line represents the average accuracy of subjects in a path starting from one of the initial 10 subjects. Each solid line represents the average across all paths per round of interaction.

2.7.3 Can non-experts improve a classifier?

If one notes that a classifier is untrustworthy, a common task in machine learning is feature engineering, i.e. modifying the set of features and retraining in order to improve generalization. Explanations can aid in this process by presenting the important features, particularly for removing features that the users feel do not generalize.

We use the 20 newsgroups data here as well, and ask Amazon Mechanical Turk users to identify which words from the explanations should be removed from subsequent training, for the worse classifier from the previous section (§2.7.2). In each round, the subject marks words for deletion after observing $B = 10$ instances with $K = 10$ words in each explanation (an interface similar to Figure 2.2, but with a single algorithm). As a reminder, the users here are not experts in machine learning and are unfamiliar with feature engineering, thus are only identifying words based on their semantic content. Further, users do not have any access to the *religion* dataset – they do not even know of its existence. We start the experiment with 10 subjects. After they mark words for deletion, we train 10 different classifiers, one for each subject (with the corresponding words removed). The explanations for each classifier

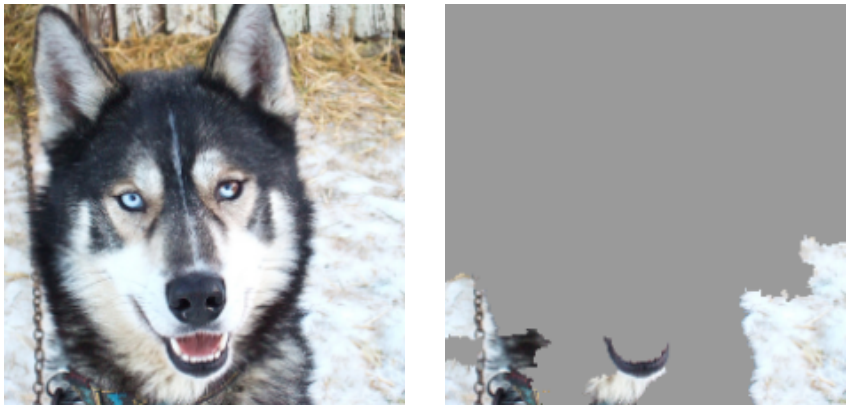
are then presented to a set of 5 users in a new round of interaction, which results in 50 new classifiers. We do a final round, after which we have 250 classifiers, each with a path of interaction tracing back to the first 10 subjects.

The explanations and instances shown to each user are produced by **SP-LIME** or **RP-LIME**. We show the average accuracy on the *religion* dataset at each interaction round for the paths originating from each of the original 10 subjects (shaded lines), and the average across all paths (solid lines) in Figure 2.10. It is clear from the figure that the crowd workers are able to improve the model by removing features they deem unimportant for the task. Further, **SP-LIME** outperforms **RP-LIME**, indicating selection of the instances to show the users is crucial for efficient feature engineering.

Each subject took an average of 3.6 minutes per round of cleaning, resulting in just under 11 minutes to produce a classifier that generalizes much better to real world data. Each path had on average 200 words removed with **SP**, and 157 with **RP**, indicating that incorporating coverage of important features is useful for feature engineering. Further, out of an average of 200 words selected with SP, 174 were selected by at least half of the users, while 68 by *all* the users. Along with the fact that the variance in the accuracy decreases across rounds, this high agreement demonstrates that the users are converging to similar *correct* models. This evaluation is an example of how explanations make it easy to improve an untrustworthy classifier – in this case easy enough that machine learning knowledge is not required.

2.7.4 Do explanations lead to insights?

Often artifacts of data collection can induce undesirable correlations that the classifiers pick up during training. These issues can be very difficult to identify just by looking at the raw data and predictions. In an effort to reproduce such a setting, we take the task of distinguishing between photos of Wolves and Eskimo Dogs (huskies). We train a logistic regression classifier on a training set of 20 images, hand selected such that all pictures of wolves had snow in the background, while pictures of huskies did not. As the features for the images, we use the first max-pooling layer of Google's pre-trained Inception neural network [87]. On a collection



(a) Husky classified as wolf **(b) Explanation**

Figure 2.11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

of additional 60 images, the classifier predicts "Wolf" if there is snow (or light background at the bottom), and "Husky" otherwise, regardless of animal color, position, pose, etc. We trained this *bad* classifier intentionally, to evaluate whether subjects are able to detect it.

The experiment proceeds as follows: we first present a balanced set of 10 test predictions (without explanations), where one wolf is not in a snowy background (and thus the prediction is "Husky") and one husky is (and is thus predicted as "Wolf"). We show the "Husky" mistake in Figure 2.11a. The other 8 examples are classified correctly. We then ask the subject three questions: (1) Do they trust this algorithm to work well in the real world, (2) why, and (3) how do they think the algorithm is able to distinguish between these photos of wolves and huskies. After getting these responses, we show the same images with the associated explanations, such as in Figure 2.11b, and ask the same questions.

Since this task requires some familiarity with the notion of spurious correlations and generalization, the set of subjects for this experiment were graduate students who have taken at least one graduate machine learning course. After gathering the responses, we had 3 independent evaluators read their reasoning and determine if each subject mentioned snow,

	Before	After
Trusted the bad model	10 out of 27	3 out of 27
Snow as a potential feature	12 out of 27	25 out of 27

Table 2.2: "Husky vs Wolf" experiment results.

background, or equivalent as a feature the model may be using. We pick the majority to decide whether the subject was correct about the insight, and report these numbers before and after showing the explanations in Table 2.2.

Before observing the explanations, more than a third trusted the classifier, and a little less than half mentioned the snow pattern as something the neural network was using – although all speculated on other patterns. After examining the explanations, however, almost all of the subjects identified the correct insight, with much more certainty that it was a determining factor. Further, the trust in the classifier also dropped substantially. Although our sample size is small, this experiment demonstrates the utility of explaining individual predictions for getting insights into classifiers knowing when not to trust them and why.

2.8 Related Work

The problems with relying on validation set accuracy as the primary measure of trust have been well studied. Practitioners consistently overestimate their model's accuracy [66], propagate feedback loops [81], or fail to notice data leaks [36]. In order to address these issues, researchers have proposed tools like Gestalt [68] and Modeltracker [1], which help users navigate individual instances. These tools are complementary to LIME in terms of explaining models, since they do not address the problem of explaining individual predictions. Further, our submodular pick procedure can be incorporated in such tools to aid users in navigating larger datasets. We emphasize that users browsing through raw data may not notice problems such as the header problem in 20 newsgroups even if aided by such tools, as

the presence of such data may seem harmless until one realizes how models exploit it.

Some recent work aims to anticipate failures in machine learning, specifically for vision tasks [6, 99]. Letting users know when the systems are likely to fail can lead to an increase in trust, by avoiding "silly mistakes" [16]. These solutions either require additional annotations and feature engineering that is specific to vision tasks or do not provide insight into why a decision should not be trusted. Furthermore, they assume that the current evaluation metrics are reliable, which may not be the case if problems such as data leakage are present. Other recent work [25] focuses on exposing users to different kinds of mistakes (our pick step). Interestingly, the subjects in their study did not notice the serious problems in the 20 newsgroups data even after looking at many mistakes, suggesting that examining raw data is not sufficient. Note that Groce et al [25] are not alone in this regard, many researchers in the field have unwittingly published classifiers that would not generalize for this task. Using LIME, we show that even non-experts are able to identify these irregularities when explanations are present. Further, LIME can complement these existing systems, and allow users to assess trust even when a prediction seems "correct" but is made for the wrong reasons.

Recognizing the utility of explanations in assessing trust, many have proposed using interpretable models [92], especially for the medical domain [11, 53, 90]. While such models may be appropriate for some domains, they may not apply equally well to others (e.g. a supersparse linear model [90] with $5 - 10$ features is unsuitable for text applications). Interpretability, in these cases, comes at the cost of flexibility, accuracy, or efficiency, as argued in Section 2.3. For text, EluciDebug [49] is a full human-in-the-loop system that shares many of our goals (interpretability, faithfulness, etc). However, they focus on an already interpretable model (Naive Bayes). In computer vision, systems that rely on object detection to produce candidate alignments [35] or attention [98] are able to produce explanations for their predictions. These are, however, constrained to specific neural network architectures or incapable of detecting "non object" parts of the images. Here we focus on general, model-agnostic explanations that can be applied to any classifier or regressor that is appropriate for the domain - even ones that are yet to be proposed.

A common approach to model-agnostic explanation is learning a potentially interpretable model on the predictions of the original model [4, 13, 79]. Having the explanation be a gradient vector [4] captures a similar locality intuition to that of LIME. However, interpreting the coefficients on the gradient is difficult, particularly for confident predictions (where gradient is near zero). Further, these explanations approximate the original model *globally*, thus maintaining local fidelity becomes a significant challenge, as our experiments demonstrate. In contrast, LIME solves the much more feasible task of finding a model that approximates the original model *locally*. The idea of perturbing inputs for explanations has been explored before [84], where the authors focus on learning a specific *contribution* model, as opposed to our general framework. None of these approaches explicitly take cognitive limitations into account, and thus may produce non-interpretable explanations, such as a gradients or linear models with thousands of non-zero weights. The problem becomes worse if the original features are nonsensical to humans (e.g. word embeddings). In contrast, LIME incorporates interpretability both in the optimization and in our notion of *interpretable representation*, such that domain and task specific interpretability criteria can be accommodated.

2.9 Conclusion

In this chapter, we argued that trust is crucial for effective human interaction with machine learning systems, and that explaining individual predictions is important in assessing trust. We proposed LIME, a modular and extensible approach to faithfully explain the predictions of *any* model in an interpretable manner. We also introduced SP-LIME, a method to select representative and non-redundant predictions, providing a global view of the model to users. Our experiments demonstrated that explanations are useful for a variety of models in trust-related tasks in the text and image domains, with both expert and non-expert users: deciding between models, assessing trust, improving untrustworthy models, and getting insights into predictions.

Chapter 3

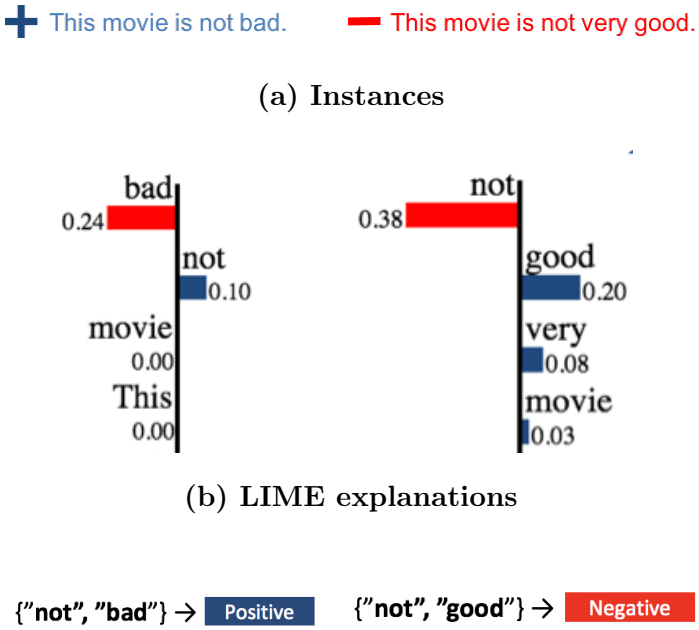
ANCHORS: HIGH-PRECISION MODEL-AGNOSTIC EXPLANATIONS

3.1 Introduction

We have been discussing how the black box model has become more prevalent with the recent successes of deep neural networks and ensembles. As a consequence of the need for users to understand the behavior of these models, *interpretable machine learning* has seen a resurgence in recent years, ranging from the design of novel *globally*-interpretable machine learning models [50, 90, 92] to local explanations (for individual predictions) that can be computed for any classifier, such as the one we presented in the previous chapter.

A question at the core of interpretability is whether humans understand a model enough to make accurate predictions about its behavior on unseen instances. For instances where humans can confidently predict the behavior of a model, let (human) *precision* be the fraction in which they are correct (note that this is human precision, not model precision). High human precision is paramount for real interpretability - one can hardly say they understand a model if they consistently think they know what it will do, but are often mistaken.

Most local approaches (including LIME) provide explanations that describe the local behavior of the model using a linearly weighted combination of the input features [4, 84]. Linear functions can capture relative importance of features in an easy-to-understand manner. However, since these linear explanations are in some way local, it may not be clear whether they *apply* to an unseen instance. In other words, their coverage (region where explanation applies) is unclear. Unclear coverage can lead to low human precision, as users may think an insight from an explanation applies to unseen instances even when it does not. When combined with the arithmetic involved in computing the contribution of the features in linear



(a) Instances

(b) LIME explanations

(c) Anchor explanations

Figure 3.1: Sentiment predictions, LSTM.

explanations, the human effort required can be quite high.

Take for example LIME explanations for two sentiment predictions made by an LSTM in Figure 3.1. Although both explanations are computed to be *locally accurate*, if one took the explanation on the left and tried to apply it to the sentence on the right, one might be tempted to think that the word "not" would have a positive influence, which it does not. While such explanations provide insight into the model, their coverage is not clear, e.g. when does "not" have a positive influence on sentiment?

In this chapter, we introduce novel model-agnostic explanations based on if-then rules, which we call *anchors*. An *anchor* explanation is a rule that sufficiently "anchors" the prediction locally – such that changes to the rest of the feature values of the instance do not matter. In other words, for instances on which the anchor holds, the prediction is (almost) always the same. For example, the anchors in Figure 3.1c state that the presence of the words

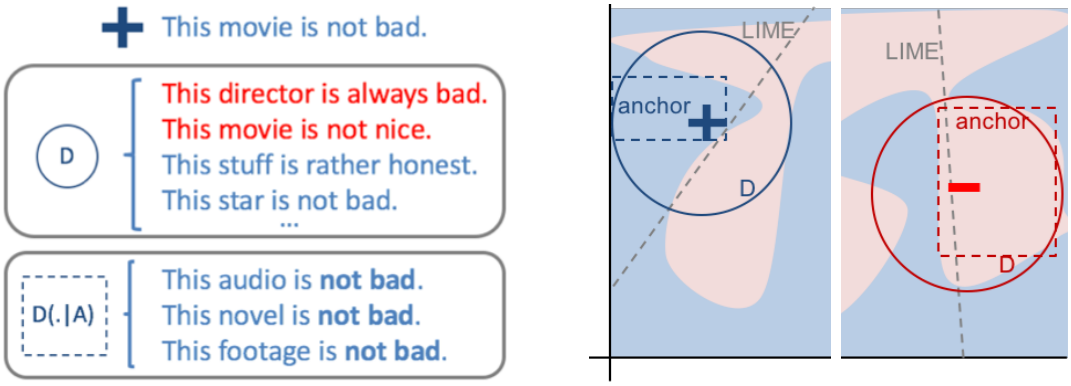
"not bad" virtually guarantee a prediction of positive sentiment (and "not good" of negative sentiment). Anchors are intuitive, easy to comprehend, and have extremely clear coverage – they only apply when all the conditions in the rule are met, and if they apply the precision is high (by design).

We demonstrate the usefulness of anchors by applying them to a variety of machine learning tasks (classification, structured prediction, text generation) on a diverse set of domains (tabular, text, and images). We also run a user study, where we observe that anchors enable users to predict how a model would behave on unseen instances with much less effort and higher precision as compared to existing techniques for model-agnostic explanation, or no explanations. We also show a trade-off between LIME and anchors: anchors allow for higher precision, while LIME produces explanations that have higher coverage.

3.2 Anchors as High-Precision Explanations

Given a black box model $f : X \to Y$ and an instance $x \in X$, the goal of local model-agnostic interpretability [75, 84] is to explain the behavior of $f(x)$ to a user, where $f(x)$ is the individual prediction for instance x. The assumption is that while the model is globally too complex to be explained succinctly, "zooming in" on individual predictions makes the explanation task feasible. Most model-agnostic methods work by perturbing the instance x according to some "perturbation distribution" $\mathcal{D}_x$ (for simplicity, $\mathcal{D}$ from now on). In Chapter 2, we emphasized that the perturbations $\mathcal{D}$ (and explanations) must use an interpretable representation (i.e. one that makes sense to humans), even if the model uses an alternative representation of the input.

Let A be a rule (set of predicates) acting on such an interpretable representation, such that $A(x)$ returns 1 if all its feature predicates are true for instance x. For example, in Figure 3.2a (top), x = "This movie is not bad.", $f(x) = \textit{Positive}$, $A(x) = 1$ where $A = \{$"not", "bad"$\}$. Let $\mathcal{D}(\cdot|A)$ denote the conditional distribution when the rule A applies (e.g. similar texts where "not" and "bad" are present, Figure 3.2a bottom). A is an *anchor* if $A(x) = 1$ and A is a sufficient condition for $f(x)$ with high probability — in our running example, if a sample z



(a) $\mathcal{D}$ and $\mathcal{D}(.|A)$ **(b) Two toy visualizations**

Figure 3.2: Concrete example of $\mathcal{D}$ in (a) and intuition (b).

from $\mathcal{D}(z|A)$ is likely predicted as *Positive* (i.e. $f(x) = f(z)$). Formally A is an anchor if,

$$\mathbb{E}_{\mathcal{D}(z|A)}[\mathbb{1}\big[f(x) = f(z)\big]] \geq \tau, A(x) = 1. \tag{3.1}$$

Figure 3.2b shows two "zoomed in" regions of a complex model, with particular instances (+ and -) being explained. LIME explanations work by learning the lines that best approximate the model under $\mathcal{D}$, with some local weighting. The resulting explanations by themselves give no indication of how faithful they are, (the explanation on the right is a much better local approximation of the black box model than the one on the left), or what their "local region" is. In contrast, even though they use the same $\mathcal{D}$, anchors are by construction faithful, adapting their coverage to the model's behavior (the anchor on the right of Figure 3.2b is broader) and making their boundaries clear.

Leaving the discussion of how to compute anchors for later, we now demonstrate their usefulness and flexibility via concrete examples in a variety of domains and models.

Text Classification: We have already alluded to Figure 3.1, where we trained an LSTM model with paraphrastic sentence embeddings [97] to predict the sentiment of reviews. In this case, the *features* used by the model are uninterpretable. The interpretable representation we

Instance	If	Predict
I want to play(V) ball.	previous word is `PARTICLE`	play is `VERB`.
I went to a play(N) yesterday.	previous word is `DETERMINER`	play is `NOUN`.
I play(V) ball on Mondays.	previous word is `PRONOUN`	play is `VERB`.

Table 3.1: Anchors for Part-of-Speech tag for the word "play".

use is the presence of individual tokens (words) in the instance. The perturbation distribution $\mathcal{D}$ replaces "absent" tokens by random words with the same POS tag with probability proportional to their similarity in an embedding space [69] (i.e. the generated sentences are coherent, and of the same length). We show samples from $\mathcal{D}$ for a sentence in Figure 3.2a. The anchor $A = \{$"not", "bad"$\}$ is easy to apply: if the words "not" and "bad" are in the sentence, the model will predict "positive" with probability at least τ (set to 0.95 from here onwards). If either word (or both) are not present, we know we do not have sufficient information to know what the model will do. Examples from $\mathcal{D}(z|A)$ are shown in Figure 3.2a (bottom).

Structured prediction: When the output of the algorithm is a structure, the anchor approach can be used to explain any function of the output. Anchors are particularly suited for structured prediction models: while the global behavior is too complex to be captured by simple interpretable models, the local behavior can usually be represented using short rules.

In Table 3.1, we explain the predictions of a state of the art part-of-speech tagger for the word "play" in different contexts, where we include the tags for neighboring words as part of the interpretable representation. The anchors demonstrate that the model is picking up on reasonable patterns of the English language, e.g. if play is preceded by a determiner, it is likely used as a noun.

In Table 3.2, we compute anchors for a multi-layer RNN encoder/attention-based decoder

English	Portuguese
This is the **question** we must address	Esta é a questão que temos que enfrentar
This is the **problem** we must address	Este é o problema que temos que enfrentar
This is what we must address	É isso que temos de enfrentar

Table 3.2: Anchors (in bold) of a machine translation system for the Portuguese word for "This" (in pink).

translation system [5] trained on English-Portuguese parallel corpora. In this case, anchors explain the presence of a word (or certain words) in the translation. The anchors (in bold) are computed with respect to the presence of the words in pink in the Portuguese text. The first row in Table 3.2 means that when the words "This", "is", and "question" appear in English, the translation will include the word "Esta". In Portuguese, the translation for the word "this" depends on the gender of the word it refers to ("esta" for feminine, "este" for masculine), or should be "isso" if its referent is not in the sentence. The anchors show that the model is capturing this behavior as it always includes "this is", and the word that "this" refers to ("question" is feminine, "problem" is masculine).

Tabular Classification: Classification of data in tabular form (categorical and/or continuous features) is a popular application of machine learning. We use a validation dataset to define $\mathcal{D}$, and sample from $\mathcal{D}(z|A)$ by fixing the predicates in A and sampling the rest of the row as a whole. We show anchors for a few predictions of 400 gradient boosted trees trained on balanced versions of three datasets in Table 3.3. The anchors provide valuable insight into these models. Marital status appears in many different anchors for predicting whether a person makes $>$ \$50K annually (*adult* dataset). The anchors for predicting recidivism for individuals released from prison [80] (*rcdv* dataset), show that this model is unfair if used for bail decisions, as race and gender feature prominently. For predicting whether a loan on the Lending Club website will turn out bad, i.e. late payment or default (*lending* dataset), the

	If	Predict
adult	No capital gain or loss, never married	$\leq$ 50K
	No capital loss, married, 37 $<$ age $<$ 48	$>$ 50K
	Country is US, married, work hours $>$ 45	$>$ 50K
rcdv	No priors, no prison violations and crime not against property	Not rearrested
	Male, black, 1 to 5 priors, not married, and crime not against property	Re-arrested
	No priors, no prison violations, white, crime not against person, and no drugs	Not rearrested
lending	FICO score $\leq$ 649	Bad Loan
	649 $\leq$ FICO score $\leq$ 699 and \$5,400 $\leq$ loan amount $\leq$ \$10,000	Good Loan
	FICO score $>$ 699	Good Loan

Table 3.3: Generated anchors for Tabular datasets.

FICO score is sufficient in the extremes, but loan amount is taken into consideration otherwise. We note that these are not exhaustive – the models are complex, and these anchors explain their behavior on part of the input space but not all of it. Anchors for "hard" predictions (in particular boundary cases) may be longer.

Image Classification: When explaining the label prediction for an image we segment the image into superpixels [91] and use the presence or absence of these superpixels as the interpretable representation. In contrast to our example in Chapter 2, instead of *hiding* superpixels, we define $\mathcal{D}(z|A)$ by fixing the superpixels in A to the original image and superimposing another image over the rest of the superpixels, which leads to more robust explanations. We explain a prediction of the Inception-V3 neural network [87] in Figure 3.3b. Even though $\mathcal{D}$ is quite unrealistic here, the anchor demonstrates that the model focuses on various parts of the dog to determine its breed. An inspection of the (admittedly bizarre) images from $\mathcal{D}(z|A)$ in Figure 3.3c for which the model predicts "beagle" with high confidence illustrates that attributes that humans would consider essential for a beagle prediction (legs, not being underwater, not being in the sky, not having a human body) are not quite as essential to the neural network.

(a) **Original image** (b) **Anchor for "beagle"** (c) **Images where Inception predicts** $P(\textbf{beagle}) > 90\%$

What animal is featured in this picture ?	**dog**
What floor is featured in this picture?	dog
What toenail is paired in this flowchart ?	dog
What animal is shown on this depiction ?	dog

(d) **VQA: Anchor (bold) and samples from** $\mathcal{D}(z|A)$

Where is the **dog**?	on the floor
What color is the **wall?**	white
When was this picture taken?	during the day
Why is he lifting his paw?	to play

(e) **VQA: More example anchors (in bold)**

Figure 3.3: Anchor Explanations for Image Classification and Visual Question Answering (VQA).

Visual Question Answering (VQA): As a final example, we present anchors for the VQA task: answering a question asked of a reference image. Here, we are interested in identifying which part of the *question* led to the predicted answer, and thus use the same

representation and distribution as in Figure 3.2b. We explain predictions from the Visual7W open-ended VQA system [105] using Figure 3.3a as the reference image. In Figure 3.3d we show a question/prediction pair and its anchor (in bold), as well as samples from $\mathcal{D}(z|A)$ and their predictions. The short anchor ("What") reveals that, for this image, the model's answer to many questions will be "dog", contrary to our expectations. In Figure 3.3e, we show other question/answer pairs and their anchors, providing examples where the behavior of the classifier is aligned with our intuitions (first three) and where it is not (last question).

3.3 Related Work

We have surveyed some of the relevant literature in Chapter 2, but we reiterate that even in the few cases where having some understanding of a machine learning model's behavior is not a requirement, it is certainly an advantage. Relying only on validation accuracy has many well studied problems, as practitioners consistently overestimate their model's accuracy [66], propagate feedback loops [81], or fail to notice data leaks [36].

Compared to other interpretable options, rules fare well; users prefer, trust and understand rules better than alternatives [55, 85], in particular rules similar to anchors. Short, disjoint rules are easier to interpret than hierarchies like decision lists or trees [50]. A number of approaches construct globally interpretable models, many based on rules [50, 53, 92, 93]. With such models, the user should be able to guess the model's behavior on any example (i.e. perfect coverage). However, these models are not appropriate for many domains, e.g. almost no interpretable rule-based system is suitable for text or image applications, due to the sheer size of the feature space, or are just not accurate enough. Interpretability, in these cases, comes at the cost of flexibility, accuracy, or efficiency (see 2.3). An alternative is learning a simple (interpretable) model to imitate the black box model globally (e.g. a decision tree [13] or a set of rules [79]), but this may yield low human precision. Simple models are not able to fully capture the behavior of the complex ones, and thus lead users to wrong conclusions, especially since it is not clear when the simple model is faithful.

To avoid this, *local* model-agnostic explanations explain individual predictions (instead of

the whole model at once). These methods provide a trade-off: each explanation is easy to understand even for complex models and tasks, but only captures the behavior of the model on a local region of the input space. The anchor approach falls within this category (and thus can explain complex models like translation and VQA), together with different forms of linear explanations including LIME [4, 74, 84]. As illustrated pictorially by Figure 3.2b and concretely in Figure 3.1b, even the local behavior of a model may be extremely non-linear, leading to poor linear approximations and users being potentially misled as to how the model will behave (e.g. thinking that "not" will usually be positive after seeing Figure 3.1b (left)). Linear explanations are also harder to parse and apply than simple rules, as they involve mental calculations. Finally, even if the black-box model is approximately linear locally, humans may not be able to compute distance functions "correctly", and may think that a local explanation applies when it does not – the "unclear coverage" problem.

Anchors, on the other hand, make their coverage very clear — the user knows exactly when the explanation for an instance "generalizes" to other instances. By construction, anchors are not only faithful to the original model, but communicate its behavior to the user in such a way that virtually guarantees correct understanding, and thus high precision. Anchors are also able to capture non-linear behavior, even locally. In sum, the anchor approach combines the benefits of local model-agnostic explanations with the interpretability of rules, constructed in a way to best support human understanding.

3.4 Efficiently Computing Anchors

We revisit the problem definition from Eq. (3.1): given a black-box classifier f, instance x, distribution $\mathcal{D}$, and the desired level of precision τ, an anchor A is a set of feature predicates on x that achieves $\text{prec}(A) \geq \tau$, where

$$\text{prec}(A) = \mathbb{E}_{\mathcal{D}(z|A)} \left[\mathbb{1}\big[f(x) = f(z)\big] \right]. \tag{3.2}$$

For an arbitrary $\mathcal{D}$ and black-box model f, it is intractable to compute this precision directly. Instead, we introduce a probabilistic definition: anchors satisfy the precision

constraint with high probability.

$$P\left(\text{prec}(A) \geq \tau\right) \geq 1 - \delta \tag{3.3}$$

If multiple anchors meet this criterion, those that describe the behavior of a larger part of the input space are preferred, i.e. ones with the largest *coverage* . Formally, we define the coverage of an anchor as the probability that it applies to samples from $\mathcal{D}$, i.e. $\text{cov}(A) = \mathbb{E}_{\mathcal{D}(z)}[A(z)]$.

We thus define this search for an anchor as the following combinatorial optimization problem:

$$\max_{A \text{ s.t. } P(\text{prec}(A) \geq \tau) \geq 1-\delta} \text{cov}(A). \tag{3.4}$$

The number of all possible anchors is exponential, and it is intractable to solve this problem exactly. While the search for anchors is similar in spirit to Probabilistic Inductive Logic Programming (ILP) [14] and other rule-finding methods, one crucial difference is that we do not assume a dataset apriori - instead we have perturbation distributions and a black box model, which we can call to estimate precision and coverage bounds under $\mathcal{D}$. While we could in theory generate a very large dataset and then use methods like ILP to find anchors, the number of perturbed samples and predictions from the black box model would be prohibitive, especially in high-dimensional sparse domains such as text. In order to efficiently explore the model's behavior in the perturbation space, we turn to a multi-armed bandit formulation.

3.4.1 Bottom-up Construction of Anchors

We first introduce a bottom-up construction of anchors, which we later extend to search over a space of potential anchors. Here, we incrementally construct the anchor A, which is initialized with an *empty* rule, i.e. one that applies to every instance. In each iteration, we generate a number of candidate rules that extend A by one additional feature predicate, $\{a_i\}$, in its definition, i.e. the set of candidate rules in each iteration is $\mathcal{A} = \{A \wedge a_i, A \wedge a_{i+1}, A \wedge a_{i+2}, \ldots\}$. We identify the candidate rule with the highest *estimated precision* (as described next), replace A with the selected candidate, and repeat. If the current candidate rule meets the anchor

definition in Eq. (3.3), we have identified our desired anchor and terminate. Although this approach does not directly compute the coverage, and instead tries to find the *shortest* anchor, we note that short anchors are likely to have a higher coverage, and require less effort from the users to understand.

In order to select the best candidate rule in each iteration, we want to estimate the precision of these candidates efficiently. Since we cannot compute the *true* precision, we rely on samples from $\mathcal{D}(\cdot|A)$ to estimate the precision of A; however, a fixed number of samples may be too many or too few for an accurate estimation. Instead, we are interested in a *minimal* set of calls to f (fewest samples from $\mathcal{D}$) in order to estimate which candidate rule has the highest *true* precision.

This problem can be formulated as an instance of *pure-exploration multi-armed bandit* problem [37], i.e. each candidate A is an arm, the true precision of A on $\mathcal{D}(\cdot|A)$ is the *latent* reward, and each pull of the arm A is an evaluation of $\mathbb{1}\big[f(x) = f(z)\big]$ on a sample from $\mathcal{D}(z|A)$. For such a setting, the KL-LUCB [37] algorithm can be used to identify the rule with the highest precision. The algorithm works by constructing confidence regions based on KL divergence [12]. In each step, the algorithm selects two distinct rules: the best mean (A) and the highest upper bound (A'), and updates their bounds by getting a sample each from $\mathcal{D}(z|A)$ and $\mathcal{D}(z'|A')$, and computing $\mathbb{1}\big[f(x) = f(z)\big]$ and $\mathbb{1}\big[f(x) = f(z')\big]$. This sampling process continues until the lower bound on A is higher than A''s upper bound with tolerance $\epsilon \in [0, 1]$. If A^* is the arm with highest true precision, the following (proved by Kaufmann and Kalyanakrishnan) holds for the true precision of the chosen rule A:

$$P(\text{prec}(A) \geq \text{prec}(A^*) - \epsilon) \geq 1 - \delta \tag{3.5}$$

Algorithm 3 presents an outline of this approach. When evaluating if the rule chosen by KL-LUCB is an anchor, we need to be confident it meets our precision criteria. Thus, if for an identified rule A, $\text{prec}_{lb}(A) < \tau$ but $\text{prec}_{ub}(A) > \tau$, we sample from $\mathcal{D}(\cdot|A)$ until either we are confident A is an anchor ($\text{prec}_{lb}(A) > \tau$) or not ($\text{prec}_{ub}(A) < \tau$).

Algorithm 3 Identifying the *Best* Candidate for Greedy

function GENERATECANDS($\mathcal{A}, c$)

$\mathcal{A}_r \leftarrow \emptyset$

for all $A \in \mathcal{A}$; $a_i \in x, a_i \notin A$ **do**

if [**then**Only high-coverage]$\text{cov}(A \wedge a_i) > c$

$\mathcal{A}_r \leftarrow \mathcal{A}_r \cup (A \wedge a_i)$ ▷ Add as potential anchor

end if

end for

return $\mathcal{A}_r$ ▷ Candidate anchors for next round

end function

function BESTCAND($\mathcal{A}, \mathcal{D}, \epsilon, \delta$)

initialize $\text{prec}, \text{prec}_{ub}, \text{prec}_{lb}$ estimates $\forall A \in \mathcal{A}$

$A \leftarrow \arg\max_A \text{prec}(A)$

$A' \leftarrow \arg\max_{A' \neq A} \text{prec}_{ub}(A', \delta)$ ▷ δ implicit below

while $\text{prec}_{ub}(A') - \text{prec}_{lb}(A) > \epsilon$ **do**

sample $z \sim \mathcal{D}(z|A), z' \sim \mathcal{D}(z'|A')$ ▷ Sample more

update $\text{prec}, \text{prec}_{ub}, \text{prec}_{lb}$ for A and A'

$A \leftarrow \arg\max_A \text{prec}(A)$

$A' \leftarrow \arg\max_{A' \neq A} \text{prec}_{ub}(A')$

end while

return A

end function

Algorithm 4 Outline of the Beam Search

function BEAMSEARCH($f, x, \mathcal{D}, \tau$)
  hyperparameters B, ϵ, δ
  $A^* \leftarrow$ **null**, $\mathcal{A}_0 \leftarrow \emptyset$ ▷ Set of candidate rules
  loop
    $\mathcal{A}_t \leftarrow$ GenerateCands($\mathcal{A}_{t-1}$, cov(A^*))
    $\mathcal{A}_t \leftarrow$ B-BestCand($\mathcal{A}_t, \mathcal{D}, B, \delta, \epsilon$) ▷ LUCB
    if $\mathcal{A}_t = \emptyset$ **then break loop**
    for all $A \in \mathcal{A}_t$ s.t. $\text{prec}_{lb}(A, \delta) > \tau$ **do**
      if cov(A) > cov(A^*) **then** $A^* \leftarrow A$
    end for
  end loop
  return A^*
end function

3.4.2 Beam-Search for Anchor Construction

Although the greedy approach we have described so far can find short anchors with the guarantee that, for each step, the choice was near optimal with high probability, it has two major shortcomings. First, due to the greedy nature of the approach, it is only able to maintain a single rule at a time (that it incrementally augments), and thus any suboptimal choice is irreversible. Second, the greedy algorithm is not directly concerned with the coverage of the anchors, and instead returns the shortest anchor that it finds. In order to address both these concerns, we extend the greedy approach to perform a beam-search by maintaining a set of candidate rules, while guiding the search to identify amongst many possible anchors the one that has the highest coverage.

The algorithm is outlined in Algorithm 4. It is similar in structure to the greedy approach, with a set of B current candidates instead of a single one. After generating all the possible candidates rules, we select the B-best candidates to keep based on the KL-LUCB approach

with multiple arms (the *Explore-m* setting). For the tolerance $\epsilon \in [0, 1]$, this version of KL-LUCB algorithm returns a set $\mathcal{A}$ of size B that is an ϵ-approximation of $\mathcal{A}^*$, with high probability.

$$P(\min_{A \in \mathcal{A}} \text{prec}(A) \geq \min_{A' \in \mathcal{A}^*} \text{prec}(A') - \epsilon) \geq 1 - \delta \tag{3.6}$$

We omit the description of KL-LUCB for this setting, but the intuition is similar to the one in the greedy approach. Further, amongst multiple anchors that we encounter, we output the one with the highest coverage, thus directly optimizing Eq. (3.4). This condition is also used for efficient pruning of the search space – we do not store any rule that has a lower coverage than that of the best anchor found so far, since the coverage of a rule can only reduce as more predicates are added. The beam-search algorithm is therefore more likely to return an anchor with a higher coverage than the one found by the greedy approach, and thus we use this algorithm for all examples and experiments.

3.4.3 Hyperparameters and Potential Issues

A rule that exactly matches x is always a valid anchor, albeit with very low coverage, and thus, is of little use. Our algorithm can always recover this anchor, and thus is guaranteed to terminate in a bounded number of iterations. In pathological cases, it is possible for KL-LUCB to require a very large number of samples from $\mathcal{D}$ in order to separate two candidate rules with a high confidence; however, this can be alleviated by increasing the tolerance ϵ, the width δ, or by setting a maximum number of samples. In practice, all explanations present in this chapter were generated in a few seconds to few minutes. We set these parameters to reasonable values, $B = 10$, $\epsilon = 0.1$, $\delta = 0.05$, and leave an analysis of the sensitivity of our approach to these for future work.

So far, we focused on computing anchors for individual predictions. In order to gain a more complete understanding of how the model works, the user needs to examine multiple explanations. Instead of randomly selecting which anchors to show to the user, we would like to identify an *optimal* set of anchors that represent this global behavior, thereby reducing

the user effort. By observing that such an objective is *submodular*, we propose an approach for *submodular-pick* (SP) in 2 that we adapt to this setting. In particular, the approach selects K anchors that cover as many instances in the validation set as possible. We use an iterative, greedy method that provides guarantees on the quality of our solution, due to the submodular nature of the optimization [45].

3.5 Experiments

We evaluate anchor explanations for complex models on a number of tasks, primarily focusing on how they facilitate accurate predictions by users (simulated and human) on the behavior of the models on unseen instances.

3.5.1 Simulated Users

For simulated users, we use the tabular datasets previously mentioned (*adult*, *rcdv* and *lending*). Each dataset is split such that models are trained with the *training* set, explanations are produced for instances in the *validation* set, and evaluated on instances in the *test* set. For each dataset, we train three different models: logistic regression (**lr**), 400 gradient boosted trees (**gb**) and a multilayer perceptron with two layers of 50 units each (**nn**). We generate both linear LIME and anchor explanations for them.

When simulating users, we compute coverage (what fraction of the instances they predict after seeing explanations) and precision (what fraction of the predictions were correct) on the complete test set. For each dataset, model, and explanation type, we compute these metrics for the explanation of each instance in the validation data. Simulating when an anchor *applies* is clear. It is not obvious, however, how real users would use LIME explanations. Ideally, they should only apply explanations to examples that are close, but it is not clear what the distance function and the threshold for "close" should be, or if users compute distances on demand. Therefore, in this section, we simulate different behaviors, and perform a study with real users in the following section.

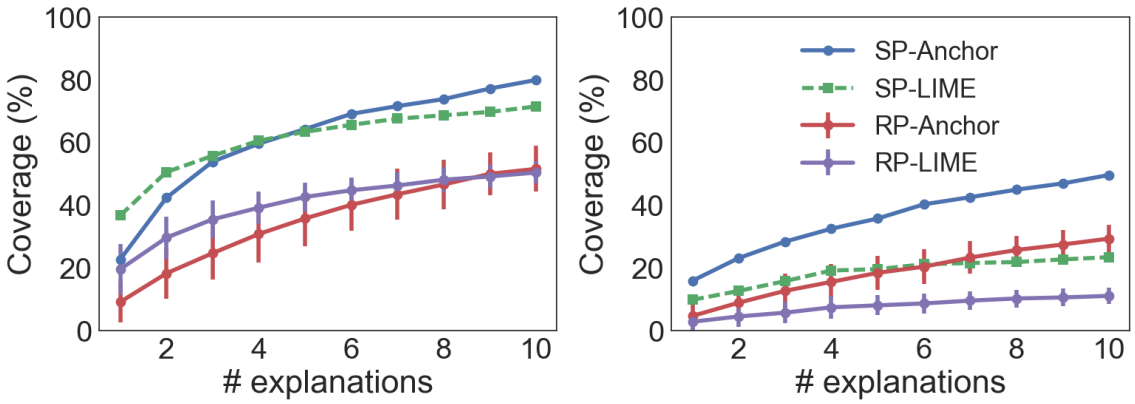
In Table 3.4 (left), we show the average precision for anchor and LIME, assuming users

		Precision		Coverage	
		anchor	**lime-n**	**anchor**	**lime-t**
adult	logistic	95.6	81.0	10.7	21.6
	gbt	96.2	81.0	9.7	20.2
	nn	95.6	79.6	7.6	17.3
rcdv	logistic	95.8	76.6	6.8	17.3
	gbt	94.8	71.7	4.8	2.6
	nn	93.4	65.7	1.1	1.5
lending	logistic	99.7	80.2	28.6	12.2
	gbt	99.3	79.9	28.4	9.1
	nn	96.7	77.0	16.6	5.4

Table 3.4: Average precision and coverage with simulated users on 3 tabular datasets and 3 classifiers. *lime-n* indicates direct application of LIME to unseen instances, while *lime-t* indicates a threshold was tuned using an oracle to achieve the same precision as the anchor approach. The anchor approach is able to maintain very high precision, while a naive use of linear explanations leads to varying degrees of precision.

always apply the LIME explanation without any regard to distance (we call such a user *lime-n*, for naive). It is clear that the anchor approach is able to deliver on the promise of high average precision, for all datasets and models. If users apply LIME naively, on the other hand, they get widely differing levels of precision. Since high precision is a prerequisite for interpretability, we simulate a user that only makes a prediction with LIME if the application of the linear explanation yields a probability above a certain threshold (setting a threshold on the distance produced strictly worse results), and call this user *lime-t*. We tune the threshold values for each dataset/model pair in order to obtain the same average precision as the anchor approach on the test set, so that coverage is comparable. A real user would not be able to perform this tuning, as (1) we are "cheating" by looking at the test set, and (2) the threshold values found range from 67% to 91%, and often with huge variation in the same dataset for different models. We show the average test coverage of the validation set explanations for anchor and lime-t in Table 3.4 (right). There is no clear winner in terms of coverage, thus demonstrating that even with the impossibly tuned thresholds, LIME is not able to outperform anchors.

Although the user is often interested in specific explanations, most users would prefer a set of explanations that explain most of the model with as little effort on their part as possible - explanations picked using the submodular procedure described before. In Figure 3.4, we show the coverage (for the same precision level) for *gb* in two of the datasets (we omit the other models/dataset due to space, but the results are similar) as the user sees more explanations, chosen either via submodular pick (SP-LIME and SP-Anchor) or at random (RP-LIME and RP-Anchor). The results indicate that while the average coverage of random explanations is low (e.g. 4.8% for *rcdv*), selecting the right set of explanations with submodular pick can give users an understanding of the model's *global* behavior (e.g. $\sim$50% after only 10 explanations). The anchor approach also yields better coverage for the same precision - even though it is unclear if real users can achieve such high precision with LIME.



(a) *adult* dataset **(b) *rcdv* dataset**

Figure 3.4: Coverage on the test set as the simulated user sees more explanations, at the same precision level. While there is no clear winner for random explanations, anchors are better when explanations are picked using submodular-pick.

3.5.2 User study

We ran a user study with 26 users – students who had or were taking a machine learning course – so we could rely on familiarity with concepts such as "model", "prediction", and "cross validation". For this study, we used the *adult* and *rcdv* datasets, followed by a multiple-choice VQA system [73] on two images. While the VQA model predicts one of 1000 labels, we restrict it to the 5 most common answers predicted on questions in $\mathcal{D}$, in order to reduce visual overload.

For each dataset and explanation type, we want to evaluate if users are able to predict the behavior of the model on unseen instances. Our set up for the *adult* and *rcdv* datasets consists of the users first browsing through 10 predictions without any explanations, then with one, and two LIME or anchor explanations. They are asked to predict the behavior of the classifier on 10 random test instances before and 10 instances after seeing each round of explanations. The user then goes through the same procedure on the other dataset, with the

explanation type that was not the one used for the first one. We ask subjects to only make predictions if they are very confident, and to select "I don't know" otherwise. We measure coverage as the fraction of instances where users made a prediction other than "I don't know" (i.e. their perceived coverage), and only measure precision in these instances. This process is repeated for the two VQA images - half the users see LIME for the first and then anchor for the second, and vice versa for the other half, and predict the model's answers on 20 questions before and after explanations. The explanations are comparable in size: LIME explanations had at most 5 terms in all datasets, while anchors varied from 1 to 5, depending on each individual prediction.

The results in Table 3.5, where LIME(1) refers to results after one LIME explanation and so on, show that users with anchors achieve high-precision - around 95% for all of the combinations, except for Anchor(2) in *adult*. The coverage of users with anchors grows with a second explanation on both datasets. Precision with LIME, on the other hand, varies depending on the dataset - although it is better than the precision with no explanations in every setting except vqa1.

The subjects made mostly correct predictions when using anchors (high precision), and knew to select "I don't know" when an instance was not covered by the explanation. In contrast, with LIME or no explanations, users thought they could make confident predictions more often, even though their precision was considerably lower. Further, it took dramatically less time to understand and use anchors as compared to linear explanations, across all datasets and tasks. On a poll, 21/26 of users preferred anchors, and 24/26 said they would be more precise with anchors; interestingly, the 2 users who said they would be more precise with LIME were actually more precise with anchors. Many commented that it was easier to apply anchors than combining the weights of LIME explanations, especially with multiple explanations. They also felt more confident in their predictions with anchors.

The user study confirms our hypotheses: it is much easier for users to understand the coverage of anchor explanations as opposed to linear explanations, and to achieve high-precision understanding of the model's behavior (as measured by predicting it on new

Method	Precision				Coverage (perceived)				Time/pred (seconds)			
	adult	rcdv	vqa1	vqa2	adult	rcdv	vqa1	vqa2	adult	rcdv	vqa1	vqa2
No expls	54.8	83.1	61.5	68.4	79.6	63.5	39.8	30.8	29.8 ±14	35.7±26	18.7±20	13.9±20
LIME(1)	68.3	98.1	57.5	76.3	89.2	55.4	71.5	54.2	28.5±10	24.6±6	8.6±3	11.1±8
Anchor(1)	100.0	97.8	93.0	98.9	43.1	24.6	31.9	27.3	13.0±4	14.4±5	5.4±2	3.7±1
LIME(2)	89.9	72.9	-	-	78.5	63.1	-	-	37.8±20	24.4±7	-	-
Anchor(2)	87.4	95.8	-	-	62.3	45.4	-	-	10.5±3	19.2±10	-	-

Table 3.5: Results of the User Study. Underline: significant w.r.t. anchors in the same dataset and same number of explanations. Results show that users consistently achieve high precision with anchors, as opposed to baselines, with less effort (time).

instances). Anchors are also easier to comprehend, and take less effort in applying, as reflected in their times and qualitative feedback. On the other hand, coverage with LIME, is generally superior to both anchors and no explanations. While both methods are superior to no explanations, there is a tradeoff between precision and coverage, and which is more important to the setting will determine the choice of explanation.

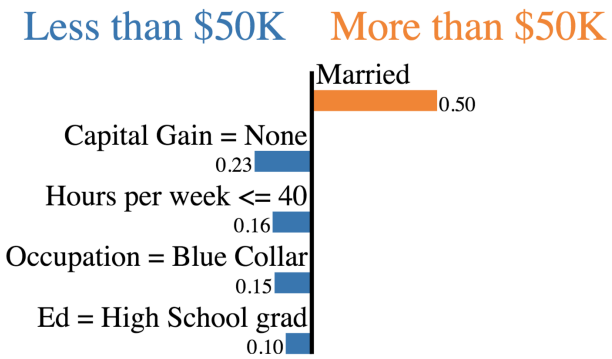
3.6 Limitations

Having demonstrated the flexibility and usefulness of the anchor approach for a wide variety of domains, and having compared it to the state-of-the-art, we now turn to its limitations and opportunities for future work.

Overly specific anchors: Predictions that are near a boundary of the black box model's decision function, or predictions of very rare classes may require very specific "sufficient conditions", and thus their anchors may be complex and provide low coverage. We give an example in Figure 3.5 (*adult* dataset), where a particular prediction is very near the boundary

$28 < \text{Age} \leq 37$
Workclass = Private
Education = High School grad
Marital Status = Married
Occupation = Blue-Collar
Relationship = Husband
Race = White
Sex = Male
Capital Gain = None
Capital Loss = Low
Hours per week ≤ 40.00
Country = United-States
$P(\textbf{Salary} > \$50\textbf{K}) = 0.57$

(a) Instance and prediction



(b) LIME explanation

IF Country = United-States **AND** Capital Loss = Low
AND Race = White **AND** Relationship = Husband
AND Married **AND** $28 < \text{Age} \leq 37$
AND Sex = Male **AND** High School grad
AND Occupation = Blue-Collar
THEN PREDICT Salary > $50K

(c) An *anchor* explanation

Figure 3.5: Explaining a prediction near the decision boundary in the UCI adult dataset.

of the decision function – almost any change to the instance results in a change in prediction. While the very specific anchor in Figure 3.5c communicates to the user that the conditions necessary for the prediction are very specific, it does not generalize well to other instances, due to its narrowness. It also does not give much insight into the model's behavior besides the fact that the instance is near a decision boundary. In cases like these, a linear LIME explanation (Figure 3.5b) is to be preferred due to the potential insights it gives, with the caveat that users may sometimes generalize incorrectly due to unclear coverage.

Potentially conflicting anchors: When using the anchor approach "in the wild", two or more anchors with different predictions may apply to the same test instance. While possible, this situation is unlikely for two reasons: (1) the high probability precision guarantee that anchors have by construction, and (2) the submodular objective in the pick procedure encourages a set of anchors with low overlap. If this were to happen in a real situation, we would want to alert the user, suggest further investigation, and maybe suggest increasing the precision threshold.

Complex output spaces: For certain problems where the output is structured and complex, there is a variety of explanations that may be useful. In this work, we restrict ourselves to explaining certain functions of the output, such as in Tables 3.1 and 3.2, leaving the task of explaining the full output space to future work. We emphasize that this is a problem that is not specific to the anchor approach, but still an open problem in the explanation literature. Even for a "simpler" output space such as the one present in the multi-label classification setting [89], it is not clear if the best option would be to explain each label individually or the set of predicted labels as a single label. The former could overwhelm the user if the number of labels is too large, while the latter may lead to non intuitive, or overly complex explanations.

3.7 Conclusions

We have argued that high precision and clear coverage are desirable properties for interpretable explanations of a model's local behavior. We introduced a novel family of rule-based, model-

agnostic explanations called *anchors*, designed to exhibit both these properties. Anchors highlight the part of the input that is sufficient for the classifier to make the prediction, making them intuitive and easy to understand. We demonstrated the flexibility of the anchor approach by explaining predictions from a variety of classifiers on domains ranging from tabular/text/image classification, translation, sequence tagging, and visual question answering. In a user study, we showed that anchors not only lead to higher human precision than LIME explanations, but also require less effort to understand and apply. On the other hand, we highlighted that a trade-off exists: LIME explanations have more coverage, and sometimes anchors are so specific that they lose their explanatory power while LIME explanations still provide insight.

Chapter 4

SEMANTICALLY EQUIVALENT ADVERSARIAL RULES FOR DEBUGGING NLP MODELS

4.1 Introduction

With increasing complexity of models for tasks like classification [34], machine comprehension [71, 82], and visual question answering [105], models are becoming increasingly challenging to debug, and to determine whether they are ready for deployment. In particular, these complex models are prone to brittleness: different ways of phrasing the same sentence can often cause the model to output different predictions.

While held-out accuracy is often useful, it is not sufficient: practitioners consistently overestimate their model's generalization [67] since test data is usually gathered in the same manner as training and validation. When deployed, these seemingly accurate models encounter sentences that are written very differently than the ones in the training data, thus making them prone to mistakes, and fragile with respect to distracting additions [32]. These problems are exacerbated by the variability in language, and by cost and noise in annotations, making such bugs challenging to detect and fix, even if one has LIME or anchor explanations.

A particularly challenging issue is *oversensitivity* [32]: a class of bugs where models output different predictions for very similar inputs. These bugs are prevalent in image classification [86], a domain where one can measure the magnitude of perturbations, and many small-magnitude changes are imperceptible to the human eye. For text, however, a single word addition can change semantics (e.g. adding "not"), or have no semantic impact for the task at hand.

Inspired by adversarial examples for images, we introduce *semantically equivalent adversaries* (**SEA**s) – text inputs that are perturbed in semantics-preserving ways, but induce

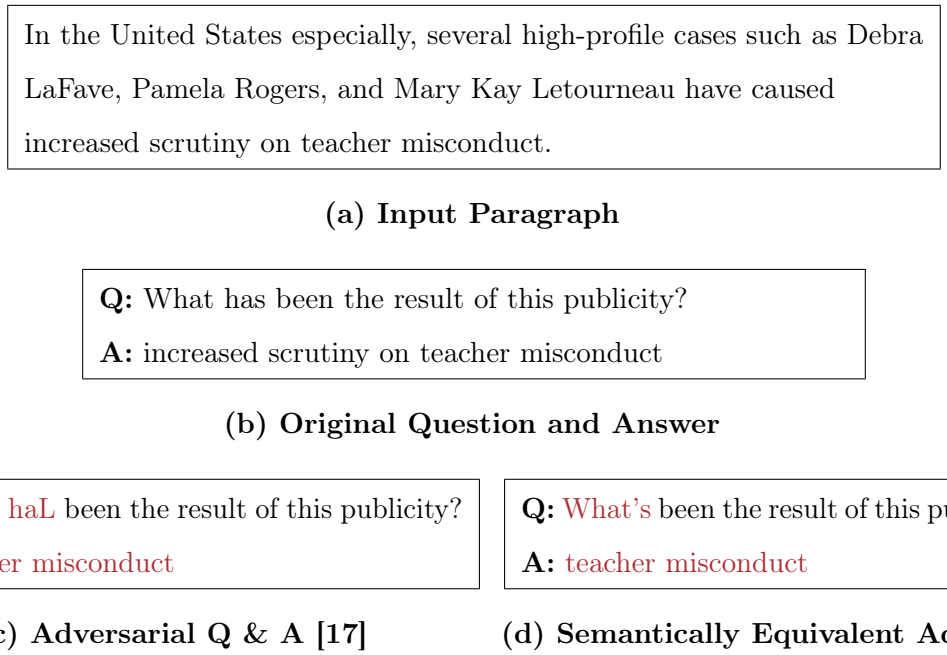


Figure 4.1: Adversarial examples for question answering, where the model predicts the correct answer for the question and input paragraph (4.1a and 4.1b). It is possible to fool the model by adversarially changing a single character (4.1c), but at the cost of making the question nonsensical. A Semantically Equivalent Adversary (4.1d) results in an incorrect answer while preserving semantics.

changes in a black box model's predictions (example in Figure 4.1). Producing such adversarial examples systematically can significantly aid in debugging ML models, as it allows users to detect problems that happen in the real world, instead of oversensitivity only to malicious attacks such as intentionally scrambling, misspelling, or removing words [7, 17, 54]. This is a different kind of explanation than the ones we presented in chapters 2 and 3, where the purpose was to understand why the model was making a prediction. Here, the purpose is to understand if the model is prone to oversensitivity bugs - if such bugs are not present in a prediction, no explanation will be produced.

While SEAs describe local brittleness (i.e. are specific to particular predictions), we are

Transformation Rules	#Flips
(*WP is→WP's*)	70 (1%)
(*?→??*)	202 (3%)

(a) Example Rules

Original: What is the oncorhynchus also called? **A:** chum salmon
Changed: What's the oncorhynchus also called? **A:** keta

(b) Example for (*WP is→WP's*)

Original: How long is the Rhine? **A:** 1,230 km
Changed: How long is the Rhine?? **A:** more than 1,050,000

(c) Example for (*?→??*)

Figure 4.2: Semantically Equivalent Adversarial Rules: For the task of question answering, the proposed approach identifies transformation rules for questions in (a) that result in paraphrases of the queries, but lead to incorrect answers (#Flips is the number of times this happens in the validation data). We show examples of rephrased questions that result in incorrect answers for the two rules in (b) and (c).

also interested in bugs that affect the model more globally. We represent these via simple replacement rules that induce SEAs on multiple predictions, such as in Figure 4.2, where a simple contraction of "is" after `Wh` pronouns (what, who, whom) (4.2b) makes 70 (1%) of the previously correct predictions of the model "flip" (i.e. become incorrect). Perhaps more surprisingly, adding a simple "?" induces mistakes in 3% of examples. We call such rules *semantically equivalent adversarial rules* (**SEAR**s).

In this chapter, we present SEAs and SEARs, designed to unveil local and global oversensitivity bugs in NLP models. We first present an approach to generate semantically equivalent adversaries, based on paraphrase generation techniques [51], that is model-agnostic (i.e. works

for any black box model). Next, we generalize SEAs into semantically equivalent rules, and outline the properties for optimal rule sets: semantic equivalence, high adversary count, and non-redundancy. We frame the problem of finding such a set as a submodular optimization problem, leading to an accurate yet efficient algorithm.

Including the human into the loop, we demonstrate via user studies that SEARs help users uncover important bugs on a variety of state-of-the-art models for different tasks (sentiment classification, visual question answering). Our experiments indicate that SEAs and SEARs make humans significantly better at detecting impactful bugs – SEARs uncover bugs that cause 3 to 4 times more mistakes than human-generated rules, in much less time. Finally, we show that SEARs are actionable, enabling the human to close the loop by fixing the discovered bugs using a data augmentation procedure.

4.2 Semantically Equivalent Adversaries (SEAs)

Consider a black box model f that takes a sentence x and makes a prediction $f(x)$, which we want to debug. We identify adversaries by generating paraphrases of x, and getting predictions from f until the original prediction is changed.

Given an indicator function $\text{SemEq}(x, x')$ that is 1 if x is semantically equivalent to x' and 0 otherwise, we define a *semantically equivalent adversary* (SEA) as a semantically equivalent instance that changes the model prediction in Eq (4.1). Such adversaries are important in evaluating the robustness of f, as each is an undesirable bug.

$$\text{SEA}(x, x') = \mathbb{1}\left[\text{SemEq}(x, x') \wedge f(x) \neq f(x')\right] \tag{4.1}$$

While there are various ways of scoring semantic similarity between pairs of texts based on embeddings [52, 96], they do not explicitly penalize unnatural sentences, and generating sentences requires surrounding context [52] or training a separate model. We turn instead to paraphrasing based on neural machine translation [51], where $P(x'|x)$ (the probability of a paraphrase x' given original sentence x) is proportional to translating x into multiple pivot languages and then taking the score of back-translating the translations into the original

language. This approach scores semantics and "plausibility" simultaneously (as translation models have "built in" language models) and allows for easy paraphrase generation, by linearly combining the paths of each back-decoder when back-translating.

Unfortunately, given source sentences x and z, $P(x'|x)$ is not comparable to $P(z'|z)$, as each has a different normalization constant, and heavily depends on the shape of the distribution around x or z. If there are multiple perfect paraphrases near x, they will all share probability mass, while if there is a paraphrase much better than the rest near z, it will have a higher score than the ones near x, even if the paraphrase quality is the same. We thus define the semantic score $S(x, x')$ as a ratio between the probability of a paraphrase and the probability of the sentence itself:

$$S(x, x') = \min\left(1, \frac{P(x'|x)}{P(x|x)}\right) \tag{4.2}$$

We define $\text{SemEq}(x, x') = \mathbb{1}[S(x, x') \geq \tau]$, i.e. x' is semantically equivalent to x if the similarity score between x and x' is greater than some threshold τ (which we crowdsource in Section 4.5). In order to generate adversaries, we generate a set of paraphrases Π_x around x via beam search and get predictions on Π_x using the black box model until an adversary is found, or until $S(x, x') < \tau$. We may be interested in the best adversary for a particular instance, i.e. $\text{argmax}_{x' \in \Pi_x} S(x, x')\text{SEA}_x(x')$, or we may consider multiple SEAs for generalization purposes. We illustrate this process in Figure 4.3, where we generate SEAs for a Visual Question Answreing model by generating paraphrases around the question, and checking when the model prediction changes. The first two adversaries with highest $S(x, x')$ are semantically equivalent, the third maintains the semantics enough for it to be a useful adversary, and the fourth is ungrammatical and thus not useful.

4.3 Semantically Equivalent Adversarial Rules (SEARs)

While finding the best adversary for a particular instance is useful, humans may not have time or patience to examine too many SEAs, and may not be able to generalize well from them in order to understand and fix the most impactful bugs. In this section, we address the

What color is the tray?	**Pink**
What colour is the tray?	Green
Which color is the tray?	Green
What color is it?	Green
How color is tray?	Green

Figure 4.3: Visual QA Adversaries: Paraphrasing questions to find adversaries for the original question (top, in bold) asked of a given image. Adversaries are sorted by decreasing semantic similarity.

problem of generalizing local adversaries into Semantically Equivalent Adversarial Rules for Text (SEARs), search and replace rules that produce semantic adversaries with little or no change in semantics, when applied to a corpus of sentences. Assuming that humans have limited time, and are thus willing to look at B rules, we propose a method for selecting such a set of rules given a reference dataset X.

A rule takes the form $r = (a \rightarrow c)$, where the first instance of the antecedent a is replaced by the consequent c for every instance that includes a, as we previously illustrated in Figure 4.2a. The output after applying rule r on a sentence x is represented as the function call $r(x)$, e.g. if $r =$(*movie*$\rightarrow$*film*), r("Great movie!") = "Great film!".

Proposing a set of rules: In order to generalize a SEA x' into a candidate rule, we must represent the changes that took place from $x \rightarrow x'$. We will use $x =$ "What color is it?" and $x' =$ "Which color is it?" from Figure 4.4 as a running example.

One approach is exact matching: selecting the minimal contiguous sequence that turns x into x', (*What*$\rightarrow$*Which*) in the example. Such changes may not always be semantics

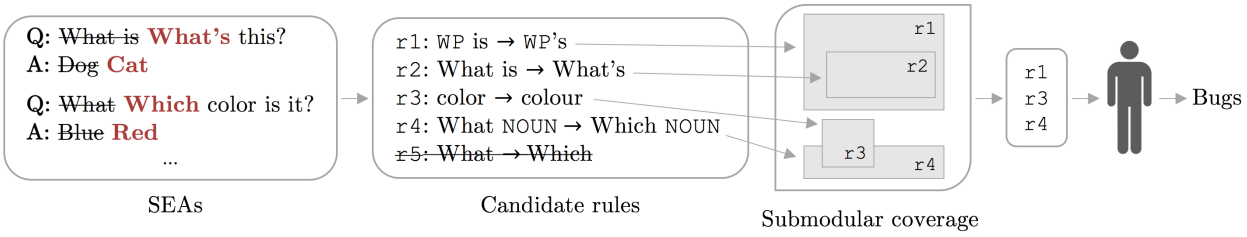


Figure 4.4: SEAR process. (1) SEAs are generalized into candidate rules, (2) rules that are not semantically equivalent are filtered out, e.g. `r5`: (*What*→*Which*), (3) rules are selected according to Eq (4.3), in order to maximize coverage and avoid redundancy (e.g. rejecting `r2`, valuing `r1` more highly than `r4`), and (4) a user vets selected rules and keeps the ones that they think are bugs.

preserving, so we also propose further rules by including the immediate context (previous and/or next word with respect to the sequence), e.g. (*What color*→*Which color*). Adding such context, however, may make rules very specific, thus restricting their value. To allow for generalization, we also represent the antecedent of proposed rules by a product of their raw text with coarse and fine-grained Part-of-Speech tags, and allow these tags to happen in the consequent if they match the antecedent. In the running example, we would propose rules like (*What color*→*Which color*), (*What* `NOUN`→*Which* `NOUN`), (`WP` *color*→*Which color*), etc.

We generate SEAs and propose rules for every $x \in X$, which gives us a set of candidate rules (second box in Figure 4.4, *for* loop in Algorithm 5).

Selecting a set of rules: Given a set of candidate rules, we want to select a set R such that $|R| \leq B$, and the following properties are met:

1. Semantic Equivalence: Application of the rules in the set should produce semantically equivalent instances. This is equivalent to considering rules that have a high probability of

inducing semantically equivalent instances when applied, i.e. $E[\text{SemEq}(x, r(x))] \geq 1-\delta$. This is the *Filter* step in Algorithm 5. For example, consider the rule (*What*→*Which*) in Fig 4.4 which produces some semantically equivalent instances, but also produces many instances that are unnatural (e.g. "What is he doing?" → "Which is he doing?"), and is thus filtered out by this criterion.

2. High adversary count: The rules in the set should induce as many SEAs as possible in validation data. Furthermore, each of the induced SEAs should have as high of a semantic similarity score as possible, i.e. for each rule $r \in R$ we want to maximize $\sum_{x \in X} S(x, r(x))\text{SEA}(x, r(x))$. In Figure 4.4, `r1` induces more and more similar mistakes when compared to $r4$, and is thus superior to `r4`.

3. Non-redundancy: Different rules in the set may induce the same SEAs, or may induce different SEAs for the same instances. Ideally, rules in the set should cover as many instances in the validation as possible, rather than focus on a small set of fragile predictions. Furthermore, rules should not be repetitive to the user. In Figure 4.4 (mid), `r1` covers a superset of `r2`'s adversaries, making `r2` completely redundant and thus not included in R.

Properties 2 and 3 combined suggest a weighted coverage problem, where a rule r covers an instance x if $\text{SEA}(x, r(x))$, the weight of the connection being given by $S(x, r(x))$. We thus want to find the set of semantically equivalent rules that:

$$\max_{R, |R|<B} \sum_{x \in X} \max_{r \in R} S(x, r(x))\text{SEA}(x, r(x)) \tag{4.3}$$

While Eq (4.3) is NP-hard, the objective is monotone submodular [46], and thus a greedy algorithm that iteratively adds the rule with the highest marginal gain offers a constant-factor approximation guarantee of $1 - 1/e$ to the optimum. This is the *SubMod* procedure in Algorithm 5, represented pictorially in Figure 4.4, where the output is a set of rules given to a human, who judges if they are really bugs or not.

Algorithm 5 Generating SEARs for a model

Require: Classifier f, Correct instances X

Require: Hyperparameters, δ, τ, Budget B

$\mathcal{R} \leftarrow \{\}$ ▷ Set of rules

for all $x \in X$ **do**

$X' = \text{GenParaphrases}(X, \tau)$

$\mathcal{A} \leftarrow \{x' \in X' \mid f(x) \neq f(x')\}$ ▷ SEAs; §4.2

$\mathcal{R} \leftarrow \mathcal{R} \cup \text{Rules}(\mathcal{A})$

end for

$\mathcal{R} \leftarrow \text{Filter}(\mathcal{R}, \delta, \tau)$ ▷ Remove low scoring SEARs

$\mathcal{R} \leftarrow \text{SubMod}(\mathcal{R}, B)$ ▷ high count / score, diverse

return $\mathcal{R}$

4.4 Illustrative Examples

Before evaluating the utility of SEAs and SEARs with user studies, we show examples in state-of-the-art models for different tasks. Note that we treat these models as black boxes, not using model internals or gradients in any way when discovering these bugs.

Machine Comprehension: We take the AllenNLP [22] implementation of BiDaF [82] for Machine Comprehension, and display some high coverage SEARs for it in Table 4.1 (also, Figures 4.1 and 4.2a). For each rule, we display two example questions with the corresponding SEA, the prediction (with corresponding change) and the percentage of "flips" - instances previously predicted correctly on the validation data, but predicted incorrectly after the application of the rule. The rule (*What* `VBZ` → *What's*) generalizes the SEA on Figure 4.1, and shows that the model is fragile with respect to contractions (flips 2% of all correctly predicted instances on the validation data). The second rule uncovers a bug with respect to simple question rephrasing, while the third and fourth rules show that the model is not robust to a more conversational style of asking questions.

SEAR	Questions / SEAs	f(x)	Flips
What VBZ → What's	~~What is~~ What's the NASUWT?	~~Trade unions~~ Teachers in Wales	2%
	~~What is~~ What's a Hauptlied?	~~main hymn~~ Veni redemptor gentium	
What NOUN → Which NOUN	~~What resource~~ Which resource was mined in the Newcastle area?	~~coal~~ wool	1%
	~~What health~~ Which health problem did Tesla have in 1879?	~~nervous breakdown~~ relations	
What VERB → So what VERB	~~What was~~ So what was Ghandi's work called?	~~Satyagraha~~ Civil Disobedience	2%
	~~What is~~ So what is a new trend in teaching?	~~Co-teaching~~ educational institutions	
What VBD → And what VBD	~~What did~~ And what did Tesla develop in 1887?	~~an induction motor~~ laboratory	2%
	~~What was~~ And what was Kenneth Swezey's job?	~~journalist~~ sleep	

Table 4.1: SEARs for Machine Comprehension.

Visual QA: We show SEARs for a state-of-the-art visual question-answering model [105] in Table 4.2. Even though the contexts are different (paragraphs for machine comprehension, images for VQA), it is interesting that both models display similar bugs. The fact that VQA is fragile to "Which" questions is because questions of this form are not in the training set,

SEAR	Questions / SEAs	f(x)	Flips
WP VBZ→	~~What has~~ What's been cut?	~~Cake~~ Pizza	3.3%
WP's	~~Who is~~ Who's holding the baby	~~Woman~~ Man	
What NOUN→	~~What~~ Which kind of floor is it?	~~Wood~~ Marble	3.9%
Which NOUN	~~What~~ Which color is the jet?	~~Gray~~ White	
color →colour	What ~~color~~ colour is the tray?	~~Pink~~ Green	2.2%
	What ~~color~~ colour is the jet?	~~Gray~~ Blue	
ADV is→	~~Where is~~ Where's the jet?	~~Sky~~ Airport	2.1%
ADV's	~~How is~~ How's the desk?	~~Messy~~ Empty	

Table 4.2: SEARs for Visual QA.

while (*color→colour*) probably stems from an American bias in data collection. Changes induced by these four rules flip more than 10% of the predictions in the validation data, which is of critical concern if the model is being evaluated for production.

Sentiment Analysis: Finally, in Table 4.3 we display SEARs for a fastText [34] model for sentiment analysis trained on movie reviews. Surprisingly, many of its predictions change for perturbations that have no sentiment connotations, even in the presence of polarity-laden words.

These examples illustrate that models are prone both to similar bugs (as in MC and VQA) and to completely different ones (sentiment analysis), which makes the detection problem much harder for humans (as our experiments in Section 4.5 will demonstrate). Practitioners understand the importance of carefully debugging models before releasing them, and these examples illustrate how SEARs can be an invaluable tool to automate part of the bug

SEAR	Reviews / SEAs	f(x)	Flips
movie → film	Yeah, the ~~movie~~ film pretty much sucked .	~~Neg~~ Pos	2%
	This is not ~~movie~~ film making .	~~Neg~~ Pos	
film → movie	Excellent ~~film~~ movie .	~~Pos~~ Neg	1%
	I'll give this ~~film~~ movie 10 out of 10 !	~~Pos~~ Neg	
is →was	Ray Charles ~~is~~ was legendary .	~~Pos~~ Neg	4%
	It ~~is~~ was a really good show to watch .	~~Pos~~ Neg	
this →that	Now ~~this~~ that is a movie I really dislike .	~~Neg~~ Pos	1%
	The camera really likes her in ~~this~~ that movie.	~~Pos~~ Neg	
`DET NOUN` is →it is	~~The movie is~~ It is terrible	~~Neg~~ Pos	1%
	~~The dialog is~~ It is atrocious	~~Neg~~ Pos	

Table 4.3: SEARs for Sentiment Analysis.

discovery process. Finally, we note that whether or not a SEAR is a bug is task-dependent: (*is*→*was*) does not change semantics for the task of sentiment analysis or VQA, but it does for MC if the paragraph discusses the past and the present. That, coupled with the hardness of paraphrase generation and evaluation, is why we assume a human will vet SEARs at the end of the process (Figure 4.4).

4.5 User Studies

We compare automatically discovered SEAs and SEARs to user-generated adversaries and rules, and propose a way to fix the bugs induced by SEARs.

Our evaluation benchmark includes two tasks: visual question answering (VQA) and

sentiment analysis on movie review sentences. We choose these tasks because a human can quickly look at a prediction and judge if it is correct or incorrect, can easily perturb instances, and judge if two instances in a pair are semantically equivalent or not. Since our focus is debugging, throughout the experiment we only considered SEAs and SEARs on examples that are originally predicted correctly (i.e. every adversary is also by construction a mistake). The user interfaces for all experiments in this section are included in the supplementary material.

4.5.1 Implementation Details

The paraphrasing model [51] requires translation models to and from different languages. We train neural machine translation models using the default parameters of OpenNMT-py [41] for English↔Portuguese and English↔French models, on 2 million and 1 million parallel sentences (respectively) from EuroParl, news, and other sources [88]. We use the spacy library (`http://spacy.io`) for POS tagging. For SEAR generation, we set $\delta = 0.1$ (i.e. at least 90% equivalence). We generate a set of candidate adversaries as described in Section 4.2, and ask mechanical turkers to judge them for semantic equivalence. Using these evaluations, we identify $\tau = 0.0008$ as the value that minimizes the entropy in the induced splits, and use it for the remaining experiments.

For VQA, we use the multiple choice *telling* system and dataset of Zhu et al. [105], using their implementation, with default parameters. The training data consists of questions that begin with "What", "Where", "When", "Who", "Why", and "How". The task is multiple choice, with four possible answers per instance. For sentiment analysis, we train a fastText [34] model with unigrams and bigrams (embedding size of 50) on RottenTomato movie reviews [65], and evaluate it on IMDB sentence-sized reviews [44], simulating the common case where a model trained on a public dataset is applied to new data from a similar domain.

4.5.2 Can humans find good adversaries?

In this experiment, we compare our method for generating SEAs with user's ability to discover semantic-preserving adversaries. We take a random sample of 100 correctly-predicted

instances for each task. In the first condition (**human**), we display each instance to 3 Amazon Mechanical Turk workers, and give them 10 attempts at creating semantically equivalent adversaries (with immediate feedback as to whether or not their attempts changed the prediction). Next, we ask them to choose the adversary that is semantically closest to the original instance, out of the candidates they generated. In the second condition (**SEA**), we generate adversaries for each of the instances, and pick the best adversary according to the semantic scorer. The third condition (**HSEA**) is a collaboration between our method and humans: we take the top 5 adversaries ranked by $S(x, x')$, and ask workers to pick the one closest to the original instance, rather than asking them to generate the adversaries.

To evaluate whether the proposed adversaries are semantically equivalent, we ask a separate set of workers to evaluate the similarity between each adversary and the original instance (with the image as context for VQA), on a scale of 1 (completely unrelated) to 5 (exactly the same meaning). Each adversary is evaluated by at least 10 workers, and considered equivalent if the median score ≥ 4. We thus obtain 300 comparisons between *human* and *SEA*, and 300 between *human* and *HSEA*.

The results in Table 4.4a and 4.4b are consistent across tasks: both models are susceptible to SEAs for a large fraction of predictions, and our fully automated method is able to produce SEAs as often as humans (left columns). On the other hand, asking humans to choose from generated SEAs (HSEA) yields much better results than asking humans to generate them (right columns), or using the highest scored SEA. The semantic scorer does make mistakes, so the top adversary is not always semantically equivalent, but a good quality SEA is often in the top 5, and is easily identified by users.

On both datasets, the automated method or humans were able to generate adversaries at the exclusion of the other roughly one third of the time, which indicates that they do not generate the same adversaries. Humans generate paraphrases differently than our method: the average character edit distance of our SEAs is 6.2 for VQA and 9.0 for Sentiment, while for humans it is 18.1 and 43.3, respectively. This is illustrated by examples in Table 4.5 - in Table 4.5a we see examples where very compact changes generate adversaries (humans were

	Human vs SEA	Human vs HSEA
Neither	145 (48%)	127 (42%)
Only Human	47 (16%)	38 (13%)
Only SEA	**54 (18%)**	**72 (24%)**
Both	**54 (18%)**	**63 (21%)**

(a) Visual Question-Answering

	Human vs SEA	Human vs HSEA
Neither	177 (59%)	161 (54%)
Only Human	45 (15%)	40 (13%)
Only SEA	**47 (16%)**	**63 (21%)**
Both	**31 (10%)**	**36 (12%)**

(b) Sentiment Analysis

Table 4.4: Finding Semantically Equivalent Adversaries: we compare how often humans produce semantics-preserving adversaries, when compared to our automatically generated adversaries (SEA, left) and our adversaries filtered by humans (HSEA, right). There are four possible outcomes: neither produces a semantic equivalent adversary (i.e. they either do not produce an adversary or the adversary produced is not semantically equivalent), both do, or only one is able to do so.

not able to find these changes though). The examples in Table 4.5b indicate that humans can generate adversaries that: (1) make use of the visual context in VQA, which our method does not, and (2) significantly change the sentence structure, which the translation-based semantic scorer does not.

Dataset	Original	SEA
VQA	Where are the men?	Where are the males?
	What kind of meat is on the boy's plate?	What sort of meat is on the boy's plate?
Sentiment	They are so easy to love, but even more easy to identify with.	They're so easy to love, but even more easy to identify with.
	Today the graphics are crap.	Today, graphics are bullshit.

(a) Automatically generated adversaries, examples where humans failed to generate SEAs (Only SEA)

Dataset	Original	Human-generated SEA
VQA	How many suitcases?	How many suitcases are sitting on the shelf?
	Where is the blue van?	What is the blue van's location?
Sentiment	(very serious spoilers) this movie was a huge disappointment.	serious spoilers this movie did not deliver what I hoped
	Also great directing and photography.	Photography and directing were on point.

(b) Human generated adversaries, examples where our approach failed to generate SEAs (Only Human)

Table 4.5: Examples of generated adversaries.

4.5.3 Can experts find high-impact bugs?

Here we investigate whether experts are able to detect high-impact global bugs, i.e. devise rules that flip many predictions, and compare them to generated SEARs. Instead of AMT workers, we have 26 expert subjects: students, graduates, or professors who have taken at

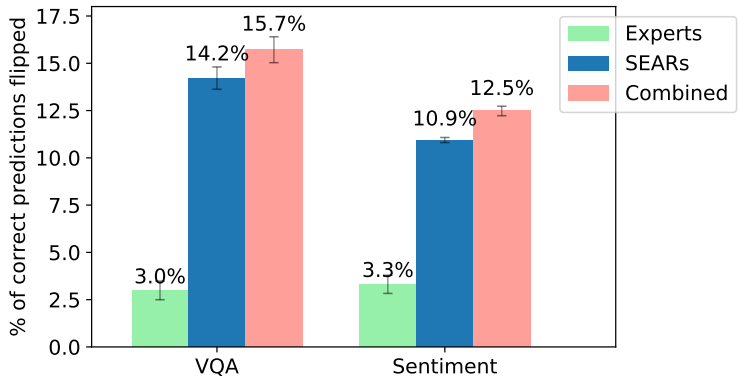


Figure 4.5: Mistakes induced by expert-generated rules (green), SEARs (blue), and a combination of both (pink), with standard error bars.

least a graduate course in machine learning or NLP[1]. The experiment setup is as follows: for each task, subjects are given an interface where they see examples in the validation data, perturb those examples, and get predictions. The interface also allows them to create search and replace rules, with immediate feedback on how many mistakes are induced by their rules. They also see the list of examples where the rules apply, so they can verify semantic equivalence. Subjects are instructed to try to maximize the number of mistakes induced in the validation data (i.e. maximize "mistake coverage"), but only through semantically equivalent rules. They can try as many rules as they like, and are asked to select the best set of at most 10 rules at the end. This is quite a challenging task for humans (yet another reason to prefer algorithmic approaches), but we are not aware of any existing automated methods. Finally, we instruct subjects they could finish each task in about 15 minutes (some took longer, some ended earlier), in order to keep the total time reasonable.

After creating their rules for VQA and sentiment analysis, the subjects evaluate 20 SEARs (one rule at a time) for each task, and accept only semantically equivalent rules. When a subject rejects a rule, we recompute the remaining set according to Eq (4.3) in real time. If a

[1]We have an IRB/consent form, and personal information was only collected as needed to compensate subjects

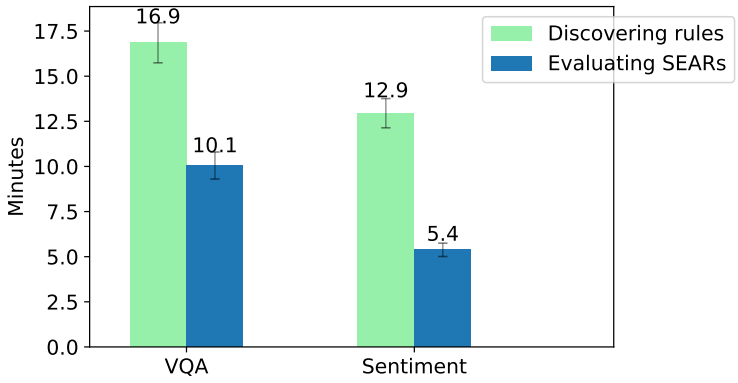


Figure 4.6: Time for users to create rules (green) and to evaluate SEARs (blue), with standard error bars.

subject accepts more than 10 rules, only the first 10 are considered, in order to ensure a fair comparison against the expert-generated rules.

We compare expert-generated rules with accepted SEARs (each subject's rules are compared to the SEARs they accepted) in terms of the percentage of the correct predictions that "flip" when the rules are applied. This is what we asked the subjects to maximize, and all the rules were ones deemed to be semantic equivalent by the subjects themselves. We also consider the union of expert-generated rules and accepted SEARs. The results in Figure 4.5 show that on both datasets, the filtered SEARs induce a much higher rate of mistakes than the rules the subjects themselves created, with a small increase when the union of both sets is taken. Furthermore, subjects spent less time evaluating SEARs than trying to create their own rules (Figure 4.6). SEARs for sentiment analysis contain fewer POS tags, and are thus easier to evaluate for semantic equivalence than for VQA.

Discovering these bugs is hard for humans (even experts) without SEARs: not only do they need to imagine rules that maintain semantic equivalence, they must also discover the model's weak spots. Making good use of POS tags is also a challenge: only 50% of subjects attempt rules with POS tags for VQA, 36% for sentiment analysis. One subject discovered

the rules (*What is→What's*), (*How is→How's*), (*Who is→Who's*), (*Where is→Where's*), (*When is→When's*), and (*Why is→Why's*), but did not think to express them more concisely with rules such as (*WP is→WP's*).

Experts accepted 8.69 rules (on average) out of 20 for VQA as semantically equivalent, and 17.32 out of 20 for sentiment analysis. Similar to the previous experiment, errors made by the semantic scorer lead to rules that are not semantically equivalent (e.g. Table 4.7). With minimal human intervention, however, SEARs vastly outperform human experts in finding impactful bugs.

4.5.4 Fixing bugs using SEARs

Once such bugs are discovered, it is natural to want to fix them. The global and deterministic nature of SEARs make them actionable, as they represent bugs in a systematic manner. Once impactful bugs are identified, we use a simple data augmentation procedure: applying SEARs to the training data, and retraining the model on the original training augmented with the generated examples.

We take the rules that are accepted by ≥ 20 subjects as accepted bugs, a total of 4 rules (in Table 4.2) for VQA, and 16 rules for sentiment (including ones in Table 4.3). We then augment the training data by applying these rules to it, and retrain the models. To check if the bugs are still present, we create a sensitivity dataset by applying these SEARs to instances predicted correctly on the validation. A model not prone to the bugs described by these rules should not change any of its predictions, and should thus have error rate 0% on this sensitivity data. We also measure accuracy on the original validation data, to make sure that our bug-fixing procedure is not decreasing accuracy.

Table 4.6 shows that the incidence of these errors is greatly reduced after augmentation, with negligible changes to the validation accuracy (on both tasks, the changes are consistent with the effect of retraining with different seeds). These results show that SEARs are useful not only for discovering bugs, but are also actionable through a simple augmentation technique for any model.

	Error rate	
	Validation	Sensitivity
Visual QA		
Original Model	44.4.%	12.6%
SEAR Augmented	45.7 %	1.4%
Sentiment Analysis		
Original Model	22.1%	12.6%
SEAR Augmented	21.3%	3.4%

Table 4.6: Fixing bugs using SEARs: Effect of retraining models using SEARs, both on original validation and on sensitivity dataset. Retraining significantly reduces the number of bugs, with statistically insignificant changes to accuracy.

4.6 Related Work

Previous work on debugging primarily focuses on *explaining* predictions in validation data in order to uncover bugs [48], or find labeling errors [101, 42]. Our work is complementary to these techniques, as they provide no mechanism to detect oversensitivity bugs. We are able to uncover these bugs even when they are not present in the data, since we generate sentences.

Adversarial examples for image recognition are typically indistinguishable to the human eye [86]. These are more of a security concern than bugs per se, as images with adversarial noise are not "natural", and not expected to occur in the real world outside of targeted attacks. Adversaries are usually specific to predictions, and even universal adversarial perturbations [64] are not natural, semantically meaningful to humans, or actionable. "Imperceptible" adversarial noise does not carry over from images to text, as adding or changing a single word in a sentence can drastically alter its meaning. Jia and Liang [32] recognize that a true analog to detect oversensitivity would need semantic-preserving perturbations, but do not pursue

an automated solution due to the difficulty of paraphrase generation. Their adversaries are whole sentence concatenations, generated by manually defined rules tailored to reading comprehension, and each adversary is specific to an individual instance. Zhao et al. [103] generate natural text adversaries by projecting the input data to a latent space using a generative adversarial networks (GANs), and searching for adversaries close to the original instance in this latent space. Apart from the challenge of training GANs to generate high quality text, there is no guarantee that an example close in the latent space is semantically equivalent. Ebrahimi et al. [17], along with proposing character-level changes that are not semantic-preserving, also propose a heuristic that replaces single words adversarially to preserve semantics. This approach not only depends on having white-box access to the model, but is also not able to generate many adversaries (only $\sim 1.6\%$ for sentiment analysis, compare to $\sim 33\%$ for SEAs in Table 4.4b). Developed concurrently with our work, Iyyer et al. [31] proposes a neural paraphrase model based on back-translated data, which is able to produce paraphrases that have different sentence structures from the original. They use paraphrases to generate adversaries and try to post-process nonsensical outputs, but they do not explicitly reject non-semantics preserving ones, nor do they try to induce rules from individual adversaries. In any case, their adversaries are also useful for data augmentation, in experiments similar to ours.

In summary, previous work on text adversaries change semantics, only generate local (instance-specific) adversaries [103, 31], or are tailored for white-box models [17] or specific tasks [32]. In contrast, SEAs expose oversensitivity for specific predictions of black-box models for a variety of tasks, while SEARs are intuitive and actionable global rules that induce a high number of high-quality adversaries. To our knowledge, we are also the first to evaluate human performance in adversarial generation (semantics-preserving or otherwise), and our extensive evaluation shows that SEAs and SEARs detect individual bugs and general patterns better than humans can.

SEAR	Questions / SEAs	f(x)
on →in	What is ~~on~~ in the background?	~~A building~~ Mountains
	What is ~~on~~? in	~~Lights~~ The television
VBP→is	Where ~~are~~ is the water bottles	~~Table~~ Vending Maching
	Where ~~are~~ is the people gathered	~~living room~~ kitchen
VERB on → VERB	What is ~~on~~ the background?	~~A building~~ Mountains
	What are the planes parked ~~on~~?	~~Concrete~~ landing strip

Table 4.7: SEARs for VQA that are rejected by users.

4.7 Limitations and Future Work

Having demonstrated the usefulness of SEAs and SEARs in a variety of domains, we now describe their limitations and opportunities for future work.

Semantic scoring errors: Paraphrasing is still an active area of research, and thus our semantic scorer is sometimes incorrect in evaluating rules for semantic equivalence. We show examples of SEARs that are rejected by users in Table 4.7 – the semantic scorer does not sufficiently penalize preposition changes, and is biased towards common terms. The presence of such errors is why we still need humans in the loop to accept or reject SEARs.

Other paraphrase limitations: Paraphrase models based on neural machine translation are biased towards maintaining the sentence structure, and thus do not produce certain adversaries (e.g. Table 4.5b), which recent work on paraphrasing [31] or generation using GANs [103] may address. More critically, existing models are inaccurate for long texts, restricting SEAs and SEARs to sentences.

Better bug fixing: Our data augmentation has the human users accept/reject rules based on whether or not they preserve semantics. Developing more effective ways of leveraging the expert's time to close the loop, and facilitating more interactive collaboration between humans and SEARs are exciting areas for future work.

4.8 Conclusion

We introduced SEAs and SEARs – adversarial examples and rules that preserve semantics, while causing models to make mistakes. We presented examples of such bugs discovered in state-of-the-art models for various tasks, and demonstrated via user studies that non-experts and experts alike are much better at detecting local and global bugs in NLP models by using our methods. We also *close the loop* by proposing a simple data augmentation solution that greatly reduced oversensitivity while maintaining accuracy. We demonstrated that SEAs and SEARs can be an invaluable tool for debugging NLP models. While different in nature from LIME or Anchors, SEAs and SEARs are complementary in how user trust can be increased, and how models can be improved with the human in the loop.

Chapter 5

IMPLICATION CONSISTENCY: EVALUATING THE REASONING CAPACITY OF MACHINE LEARNING MODELS

5.1 Introduction

With impressive results in perception tasks like image recognition [27, 87], machine learning models are now being applied to more complex tasks. There is an increasing interest in tasks that are considered close to "AI-complete", the definition of which usually include (but is not limited to) possessing general intelligence, being able to reason and being able to use data from multiple domains (e.g. image, text). Really solving the task of machine comprehension [71] requires being able to understand who the actors and actions in a paragraph are, what relationships are presented between them, which actors are being referenced by pronouns, understanding the information need posed by the question and how it can be answered, amongst other hard tasks. Similarly, the Visual Question Answering [3] task requires not only perception abilities (fine-grained recognition, object detection), but also "higher level thinking" such as how the question is related to the information presented in the image, commonsense reasoning, knowledge based reasoning, understanding of location / color / size attributes, amongst others.

Despite the apparent hardness of these tasks, current evaluation is far from being able to measure true intelligence. For example, Weissenborn et al. [94] show that models can do well in machine comprehension in the SQuAD dataset by learning context and type-matching heuristics. Similarly, various biases have been observed in the most popular visual question answering dataset, such as the fact that answering questions starting with "Do you see a ..." with "yes" results in 87% accuracy, or that "tennis" is the correct answer for 41% of questions starting with "What sport is". These findings resulted in new versions of both datasets:

SQuAD 2.0 [72] introduces questions that are not answered by the context, for which the correct answer is "unanswerable", which should thwart models relying on simple heuristics. VQA v2 [24] takes image-question pairs and adds a new image with a different answer for the same question, which thwarts models that rely too much on the question and do not pay attention to the image.

These efforts are laudable, and certainly result in better evaluation and an advancement of the field. However, they do not address a fundamental question in the evaluation of models that purport to be intelligent: it is not only individual predictions that matter, but also relationships between predictions. We have noted before how models that seem to be solving these problems can be inadequate even when making right predictions by being too stable (e.g. when anchors do not correspond to human intuition in Chapter 3) or overly sensitive (e.g. when non-semantic changes make models change predictions in Chapter 4). These examples illustrate that we instinctively care not only that a model predicts an individual example correctly, but also about the relationship between that prediction and the prediction of examples related to it (i.e. if we change the semantics of the example the prediction should change, and if we change the example in insignificant ways it should remain the same). A further example of important relationships between predictions is logical consistency, a violation of which is illustrated in Figure 5.1, where an answer of "6" in the question "How many jets?" posed in 5.1b should imply an answer of "yes" to the question "Are there 6 jets?" (Figure 5.1c).

In this chapter, we present preliminary work on a new kind of evaluation metric called Implication Consistency (Consistency, for short), which captures the intuition that certain model predictions have measurable implications for other predictions. Implication Consistency is defined over an implication graph, where nodes are $(instance, label)$ pairs, and edges signify implication relations. These are general enough to allow human stakeholders to define edges based on common sense, business rules, prior domain knowledge and etc. We emphasize that Implication Consistency is not meant to replace accuracy (and related metrics), but to be used in conjunction with it.

(a) Input image

Q: How many jets?
A: 6

(b) Question and Answer from state of the art model [102] specialized in counting

Q: Are there 6 jets?
A: no

(c) Different but related question, where the answer from the same model is inconsistent

Figure 5.1: Example of inconsistency between model outputs.

We focus in particular on Implication Consistency for question answering, where our goal is to measure the logical consistency of predictions. We give examples in Visual Question Answering and Machine Comprehension, and propose a set of programmatically created logical implications for Visual Question Answering, which can be used to evaluate any model. We show that the edges we create really are logical implications as judged by humans, and evaluate state of the art models for consistency in both the VQA and VQA v2 dataset.

5.2 Related Work

Many recognize that aggregate measures like accuracy are not sufficient for the evaluation of intelligent systems, since datasets have their own particular distributions that can easily skew results. A common practice in the field is to segregate held out datasets into semantically meaningful slices: for example, Visual Question Answering [3] evaluation usually involves reporting metrics for all questions in conjunction with metrics for questions within the

[Yes/No, Number, Other] categories. Similary, SQuAD is partitioned according to answer types: *Date, Other Numeric, Person, Location, etc.* Some propose automatically generated diagnostic datasets: bAbI [95] for Machine Comprehension and CLEVR [33] for VQA are examples, both automatically generated. While these recognize that AI intelligence must be measured over different capabilities, the evaluation is still restricted to individual units and thus cannot capture inconsistencies like predicting that an object is at the same time to the left and to the right of another object.

In an effort to eliminate language priors that shadow the role of visual understanding, the balanced abstract scene VQA dataset [100] has complementary images for each question, with opposing binary answers. They measure how often a model answers each pair correctly, with the assumption that if a model answers only one instance in the pair correctly it has not really understood the scene. This is one of the rare examples in the literature where the relationship between different predictions is measured, although the relationship is not as intuitive, and thus does not allow humans to easily grasp in which ways the model is failing. In contrast, even though VQA v2 [24] has complementary images, the evaluation still only focuses on individual predictions.

The relation prediction / extraction tasks (a system reads documents and tries to predict or extract relations between entities) seems very amenable to the kind of evaluation we are proposing: the task in itself involves making predictions on graphs, and humans have a wealth of background knowledge about many of the relations of interest. There are several attempts to leverage logical rules, e.g. extracting and combining them with matrix factorization [79] or knowledge graphs [26]. These focus on trying to learn rules like $has\_part(x, y) \rightarrow part\_of(y, x)$ in order to improve overall accuracy, but do not try to evaluate models with human-derived or approved rules. Closer to our work is the definition of InconsistencyLoss [62], which adds a penalty term in the overall loss function when the rules of Guo et al. [26] are violated. Again, the purpose of these is to leverage logic to increase overall accuracy, and the authors do not evaluate any of the models for inconsistency even in accepted logical implications like the transitivity of is_a relations - evaluation is still focused on accuracy of

individual predictions.

In summary, while there seems to be a consensus that aggregate accuracy may be misleading when measuring the intelligence of models, previous work mainly tries to address this by creating new, more diverse datasets. We propose a complementary solution: creating new metrics that take the relationship between predictions into account in ways that are intuitive and defined by humans.

5.3 Implication Consistency

Formally, we are given a dataset (X, Y), where (x_i, y_i) denotes the ith instance and the ith label, respectively. Given a model f, traditional metrics typically involve an average over each instance, under the assumption that the prediction of each instance has no bearing on other instances. Accuracy, for example, is typically defined as $\frac{\sum_i \mathbb{1}\big[f(x_i) = y_i\big]}{|X|}$. For Implication Loss, we define a directed graph $G = (V, E)$ over the dataset, where each node is an instance with associated label (x_i, y_i). An edge $(i, j) \in E$ indicates that $(x_i, y_i) \rightarrow (x_j, y_j)$ (arrow indicating implication) - that is, a label y_i for x_i implies that y_j is the label for x_j. Implication Consistency is then defined as the fraction of implications the model is able to capture out of the instances predicted correctly. Formally,

$$Consistency = \frac{\sum_{(i,j)\in E} \mathbb{1}\big[f(x_i) = y_i\big]\mathbb{1}\big[f(x_j) = y_j\big]}{\sum_{(i,j)\in E} \mathbb{1}\big[f(x_i) = y_i\big]} \tag{5.1}$$

The implications represented by edges are defined by human users, depending on the evaluation desiderata. One specific type of edge we focus on in this chapter is *Logical Consistency* implications - that is, violating an edge in the graph means the model is not following the rules of logic, something we want to measure in closer to "AI-complete" tasks like question answering. An example is presented in Figure 5.2, where we illustrate edges coming from the instance in Figure 5.1. We label the edges with sub types – one may wish to do so in order to report a summary statistic for each type as well as a general aggregate, just as in traditional accuracy metrics.

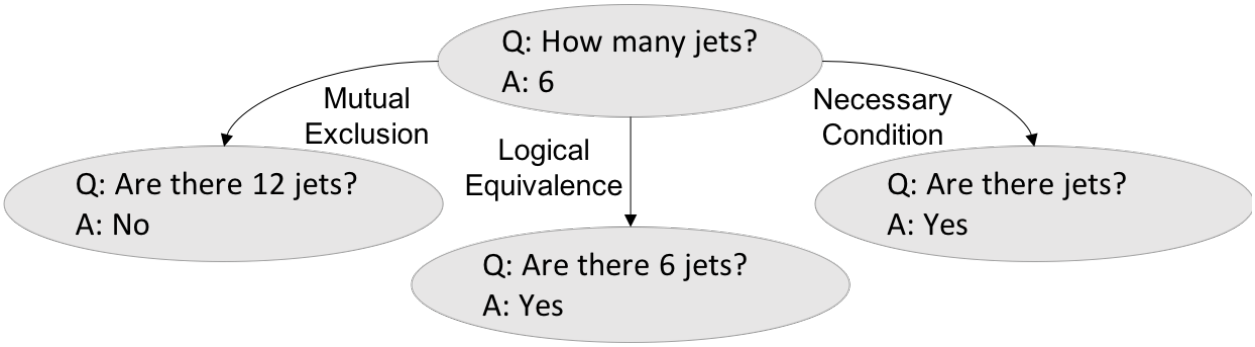


Figure 5.2: An example Consistency graph, where an edge $a \rightarrow b$ means a implies b. Edges are labeled with the type of logical implication they represent.

While we focus on logical consistency implications, Consistency is more general, and edges can represent other implication properties that are desirable by human stakeholders, such as:

- **Robustness:** an edge (i, j) exists between two examples if x_i and x_j are semantically equivalent, which in turn implies $y_i = y_j$. The metric we used for evaluating bugs induced by SEARs in Chapter 4 is an example of Implication Consistency measuring robustness.

- **Fairness:** given a dataset X, Y, one can create a copy X' where all else is equal except for protected features such as race. Edges then are created such that $(x_i, y_i) \rightarrow (x'_i, y_i)$, i.e. Consistency measures how often the prediction is not directly affected by protected features.

New nodes and edges may be added to current datasets by means of crowdsourcing, automatically via programs (as we do in the next section), or via a collaboration between the two, as we did with SEARs in Chapter 4 (rules are generated automatically, but filtered by humans). While the next section focuses on Visual Question Answering, Figure 5.3 illustrates that the same type of edges can be adapted to Machine Comprehension, albeit with more human help, as the SQuAD dataset requires answers to be present in the context paragraph.

The Rhine is a European river that begins in the Swiss canton of GraubÃijnden in the southeastern Swiss Alps, forms part of the Swiss-Austrian, Swiss-Liechtenstein border, Swiss-German and then the Franco-German border, then flows through the Rhineland and eventually empties into the North Sea in the Netherlands. The biggest city on the river Rhine is Cologne, Germany with a population of more than 1,050,000 people. It is the second-longest river in Central and Western Europe (after the Danube), at about 1,230 km (760 mi), with an average discharge of about 2,900 m3/s (100,000 cu ft/s).

(a) Input Paragraph

Q: How long is the Rhine?
A: 1,230km (760 mi)

(b) Original Question and Answer

Q: What river is about 1,230km?
A: Danube

(c) Different question where model is inconsistent

Figure 5.3: Example of inconsistency between model outputs for SQuAD (model is BiDaF [82], as implemented by AllenNLP [22]).

5.4 Logical Implications for Visual Question Answering

Models for the Visual Question Answering task have to become proficient at understanding images, questions, and how questions relate to images. Since the task allows for open-ended questions and answers, models can be checked for logical consistency – which is a way to check how much understanding the model really has about a specific question, i.e. if the model has a good grasp of the concepts and is able to do high level reasoning. For example, while the answer for the question in Figure 5.1b is impressive, Figure 5.1c reveals that the model did not grasp the concepts involved as well as Figure 5.1b would seem to indicate.

We create new datasets with associated graphs from the original VQA and VQA2 datasets,

by combining the original questions and answers and transforming them into yes/no questions for which the answer is implied by the original. In order to do this, we take the original text and fine-grained part-of-speech tag predictions from spacy [30], and create new questions via manually defined rules. For example, one of our rules verifies if both the question and answer contain a word tagged with "VBG" (Verb, gerund or present participle, e.g. swimming). If so, it asks the model "Is there anyone or anything ANS in the picture?", where *ANS* is a placeholder for the original answer. This rule assumes that if the answer is an action, some actor must be performing it, and thus is named "Action in answer needs actor". The original VQA answers are shortened and often missing prepositions or articles (e.g. "Where is the book? on table"), and thus we train a 4-gram language model with Kneser-Ney smoothing using the kenlm package [28] in order to smooth our created questions by adding prepositions, articles and changing from singular to plural, 1st to 3rd person or vice versa when appropriate. Some rules rely on antonyms, which we get from WordNet [61]. We split the rules into the following subcategories:

Logical equivalence: rules that transform the question / answer pair into a yes-no equivalent such that the answer is always yes. The rules are shown, with examples in Table 5.1, as well as the coverage they have in each dataset, i.e. how many nodes in the original dataset now have an edge of this type.

Necessary condition: rules that check for conditions that must be present for the original question and answer to make sense, presented in Table 5.2 . Violations of these seem particularly egregious - for example, a model that says there are 2 books on the table *and* that there are no books on the table has clearly failed to grasp an important concept in counting objects.

Mutual Exclusion: rules that transform the question / answer pair into yes-no equivalent such that the answer is always no, presented in Table 5.3. We leverage the crowdsourcing

performed in gathering the VQA 2.0 dataset [24], where each question is paired with two images with different (and thus mutually exclusive) answers by asking the logical equivalence questions created for one image to the paired image, expecting a negative answer. These are denoted by the postfix "inversion", and do not apply to VQA 1.0.

Overall, with a total of 13 rules for VQA 1.0 and 21 for VQA 2.0, we are able to achieve a total coverage of 60.2% and 58.8%, respectively, with many nodes being checked for consistency by multiple rules. In the next section, we evaluate the quality of these rules, as well as how well state-of-the-art models fare as evaluated by Consistency in the induced graph.

			Coverage	
Name	Example	Implication	VQA1	VQA2
Color		**Q**: What color is the motorcycle? **A**: red ↓ **Q**: Is the motorcycle red? **A**: yes	8.9%	8.3%
Number		**Q**: How many animals are here? **A**: 2 ↓ **Q**: Are 2 animals here? **A**: yes	8.7%	8.9%

Table 5.1 – *Continues in next page*

Table 5.1 – *Continued from previous page*

What is `ADP`		**Q**: What is on the hot dog? **A**: mustard ↓ **Q**: Is there mustard on the hot dog? **A**: yes	2.6%	2.5%
What is `DET`		**Q**: What is the dog riding? **A**: skateboard ↓ **Q**: Is the dog riding the skateboard? **A**: yes	8.9%	8.6%
What is `VERB`		**Q**: What is parked? **A**: car ↓ **Q**: Is the car parked? **A**: yes	1.3%	1.3%
Where		**Q**: Where is the fork? **A**: on plate ↓ **Q**: Is the fork on the plate? **A**: yes	3.3%	3.0%
		Total	33.2%	32.6%

Table 5.1: Logical Equivalence Rules

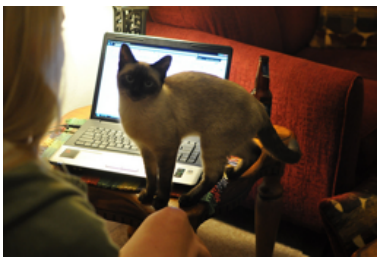
Name	Example	Implication	Coverage VQA1	Coverage VQA2
Action in answer needs actor		**Q**: What is the cat doing? **A**: watching tv ↓ **Q**: Is there anyone or anything watching tv in the picture? **A**: yes	2.0%	2.1%
Action in question needs actor		**Q**: Is he wearing glasses? **A**: yes ↓ **Q**: Is there anyone or anything wearing glasses in the picture? **A**: yes	4.6%	3.7%
Color answer in picture		**Q**: What color is the couch? **A**: red ↓ **Q**: Is there anything red in the picture? **A**: yes	8.9%	8.8%
How many > 0 implies any		**Q**: How many books are on the table? **A**: 2 ↓ **Q**: Are there any books on the table? **A**: yes	8.4%	8.4%

Table 5.2 – *Continues in next page*

Table 5.2 – *Continued from previous page*

Noun answer in picture		**Q**: What type of bird is this? **A**: seagull ↓ **Q**: Is there a seagull in the picture? **A**: yes	30.0%	29.3%
		Total	53.3%	51.6%

Table 5.2: Necessary Condition Rules

Name	Example	Implication	Coverage VQA1	Coverage VQA2
(`ADJ`, opposite) exclusion		**Q**: Are these veggies raw? **A**: yes ↓ **Q**: Are these veggies cooked? **A**: no	4.2%	3.9%
(Answer, opposite) exclusion		**Q**: What is the weather? **A**: cold ↓ **Q**: Is the weather hot? **A**: no	0.4%	0.5%

Table 5.3 – *Continues in next page*

Table 5.3 – *Continued from previous page*

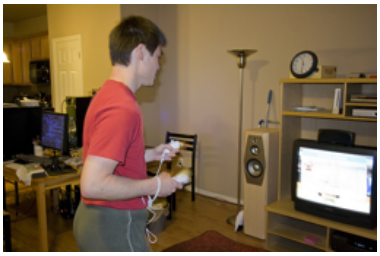
(N, N+1) exclusion		**Q**: How many benches? **A**: 2 ↓ **Q**: Are there 3 benches? **A**: no	8.7%	7.6%
(color1, color2) exclusion		**Q**: What color are her boots? **A**: red ↓ **Q**: Are her boots green? **A**: no	8.9%	7.9%
Color inversion		**Q**: What color is the hat? **A**: red ↓ **Q**: Is the hat brown? **A**: no	-	7.8%
Number inversion		**Q**: How many people are in the picture? **A**: 1 ↓ **Q**: Are 4 people in the picture? **A**: no	-	8.7%

Table 5.3 – *Continues in next page*

Table 5.3 – *Continued from previous page*

What is ADP inversion		**Q**: What is in the box? **A**: pizza ↓ **Q**: Are there donuts in the box ? **A**: no	-	2.3%
What is DET inversion		**Q**: What is the pizza in? **A**: plate ↓ **Q**: Is the pizza in a pan? **A**: no	-	7.9%
What is VERB inversion		**Q**: What is flying? **A**: plane ↓ **Q**: Are there kites flying? **A**: no	-	1.1%
Where inversion		**Q**: Where is the panda sitting? **A**: ground ↓ **Q**: Is the panda sitting on a tree? **A**: no	-	2.5%
		Total	22.3%	35.8%

Table 5.3: Mutual Exclusion Rules

5.5 Experiments

5.5.1 Validating logical consistency rules

In order to use the induced graph proposed in the previous section for evaluation, we must ascertain whether the new nodes and edges created programmatically are really logical implications of the original nodes. As a proxy for measuring both the correctness of the rules and the coherence of generated questions, we devise the following experiment. Given an original (question, answer) pair (q, a) and a generated pair (q', a') such that $(q, a) \rightarrow (q', a')$, we show subjects on Amazon Mechanical Turk (q, a) as context *without the accompanying image* and ask them q', after telling them to assume that answer a is correct. If $(q, a) \rightarrow (q', a')$, subjects should be able to answer q' correctly even in the absence of the image.

Since all generated questions are yes/no questions, we give subjects three options: "yes", "no", and "I don't know", and explicitly tell them to answer "I don't know" if the question does not make sense, or if the answer is not implied from the original (q, a) pair. We take a random sample of 100 questions covered by each rule, and ask 5 different turkers each question, taking the majority answer as ground truth.

The results for each rule and averages are presented in Table 5.4, where we demonstrate that all of our proposed rules have very high agreement with human judgment. We did not evaluate the inversion rules for VQA 2.0, as they are generated in the same way as the logical equivalence rules. With this, we conclude that Consistency over the graph induced in the previous section is a metric of logical consistency that agrees with human judgment.

5.5.2 Evaluation of state-of-the-art Visual Question Answering models with Consistency

We evaluate two models for logical consistency: the model of Kazemi and Elqursh [38], from now on denoted as *AttendVQA*, state-of-the-art on both VQA and VQA v2 in April of 2017, and the same model with an added module created specifically for counting objects [102] (*CountingVQA* from now on), improving the performance on questions with numerical components or answers and thus becoming the state of the art in February of 2018. We use a

	Rule	Turker response matches rule
Logical Equivalence	Color	100%
	Number	99%
	What is `ADP`	100%
	What is `DET`	99%
	What is `VERB`	99%
	Where	99%
	Average	**99.2**%
Necessary Condition	Action in answer needs actor	97%
	Action in question needs actor	96%
	Color answer in picture	99%
	How many > 0 implies any	99%
	Noun answer must be in picture	94%
	Average	**97**%
Mutual Exclusion	(`ADJ`, Opposite) exclusion	97%
	(Answer, Opposite) exclusion	97%
	(N, N + 1) exclusion	100%
	(color1, color2) exclusion	100%
	Average	**98.5**%
Average of all rules		**98.2**%

Table 5.4: Agreement between subjects' response on Mechanical Turk and the expected answer from our rules.

	AttendVQA		CountingVQA	
	VQA	VQA v2	VQA	VQA v2
Yes/No	72.6	59.3	71.7	70.8
Number	28.2	24.4	37.2	40.6
Other	45.4	40.8	50.5	48.6
Average	**53.3**	**45.6**	**56.8**	**55.9**

Table 5.5: Accuracy on exact match answers for models on VQA 1.0 and 2.0.

pretrained version of AttendVQA[1], trained on the VQA dataset, and a pretrained version of CountingVQA on the VQA v2 dataset [2]. The exact match accuracy of both models on both datasets is presented in Table 5.5, where CountingVQA is shown to be more accurate in general and in every category except Yes/No questions in the original VQA dataset.

We show the results of measuring Implication Consistency for both datasets and models in Table 5.6. It is interesting to note that while AttendVQA has much lower accuracy on VQA v2, the Consistency numbers are very close, with a large drop in total average caused mainly by the inversion Mutual Exclusion questions in VQA v2. In contrast, CountingVQA has similar accuracy in both datasets, but a larger drop in Consistency overall in VQA v2. While CountingVQA is more consistent on average, it is less consistent in most Necessary Condition rules, excepting the one to do with numbers.

While CountingVQA is much more accurate on Number questions, it is not much more consistent in $(N, N+1)$ mutual exclusion, and is less consistent on Number inversions. We show examples of CountingVQA inconsistency for all numerical-type edges in Figures 5.4, 5.5, 5.6 and 5.7. Detecting these types of problems allows designers to take steps to make models

[1] `https://github.com/Cyanogenoid/pytorch-vqa`

[2] `https://github.com/Cyanogenoid/vqa-counting`

		AttendVQA		CountingVQA	
		VQA	VQA v2	VQA	VQA v2
Logical Equivalence	Color	60.3	58.6	84.4	82.0
	Number	69.9	63.9	77.3	71.1
	What is `ADP`	93.8	93.3	92.2	89.0
	What is `DET`	64.2	64.2	82.2	79.8
	What is `VERB`	64.4	60.3	77.4	70.9
	Where	71.5	69.2	85.9	82.6
	Average	**65.9**	**63.8**	**82.5**	**78.7**
Necessary Condition	Action in answer needs actor	69.0	71.8	74.3	74.9
	Action in question needs actor	81.8	80.9	70.6	69.6
	Color answer in picture	92.5	92.7	74.9	69.8
	How many > 0 implies any	59.8	56.6	92.4	89.8
	Noun answer must be in picture	85.8	84.9	82.9	80.6
	Average	**83.3**	**82.5**	**80.6**	**78.3**
Mutual Exclusion	(`ADJ`, Opposite) exclusion	51.5	51.0	55.2	55.8
	(Answer, Opposite) exclusion	56.6	58.6	62.5	64.0
	(N, N + 1) exclusion	34.0	35.3	34.1	41.1
	(color1, color2) exclusion	60.5	59.2	75.4	77.7
	Color inversion	–	47.3	–	64.3
	Number inversion	–	52.4	–	38.6
	What is `ADP` inversion	–	12.9	–	51.6
	What is `DET` inversion	–	46.6	–	56.6
	What is `VERB` inversion	–	19.9	–	49.7
	Where inversion	–	45.4	–	54.3
	Average	**51.7**	**48.3**	**58.9**	**57.3**
	Total Average	**71.7**	**65.3**	**76.2**	**70.2**

Table 5.6: Implication Consistency results for both models and datasets.

more intelligent. In this case, a quick investigation indicates that roughly 0.3% of VQA and VQA v2 questions contain any numbers, while 13% of answers contain numbers and 11% of questions start with "How Many". This training data results in models that are able to count to some extent, but not able to abstract such knowledge and reason about numbers when these are part of the question.

Q: How many pieces are missing?
A: 1

Q: Is 1 piece missing?
A: no

Q: How many men?
A: 2

Q: Are there 2 men?
A: no

Figure 5.4: Examples of CountingVQA inconsistency for Logical Equivalence edges of type Number.

Q: How many propellers?
A: 2

Q: Are there any propellers?
A: no

Q: How many knives are visible?
A: 1

Q: Are there any knives visible?
A: no

Figure 5.5: Examples of CountingVQA inconsistency for Necessary Condition edges of type "How many > 0 implies any".

Q: How many people are in the photo?
A: 1

Q: Are 2 people in the photo?
A: yes

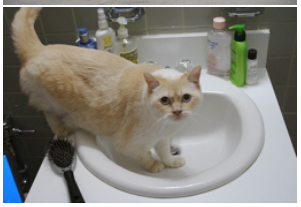
Q: How many birds are on the branch?
A: 2

Q: Are 3 birds on the branch?
A: yes

Figure 5.6: Examples of CountingVQA inconsistency for Mutual Exclusion edges of type "(N, N + 1) exclusion".

Q: How many people are riding bikes?
A: 1

Q: Are 6 people riding bikes?
A: yes



Q: How many cats?
A: 1

Q: Are there no cats?
A: yes

Figure 5.7: Examples of CountingVQA inconsistency for Mutual Exclusion edges of type "Number inversion".

5.6 Conclusion

Evaluation of intelligent systems needs to take into account relationships between outputs rather than just outputs in isolation, as true intelligence goes beyond individual predictions and involves concepts like consistency, robustness and higher level thinking.

In this chapter, we presented preliminary work on a new kind of graph-based metric to

measure such relationships, and showed examples of its application for logical consistency checking in VQA. Our evaluation revealed that even when models make the correct prediction, in many cases they are far from really understanding the concept behind the question or being able to reason about it. Such examples also give pointers to how models can be improved, which we exemplified by discovering a discrepancy in the frequency of questions for which the answer is numbers and the frequency of questions where numbers are mentioned in the VQA datasets.

While our examples relied on programmatically created edges (validated by human users), nothing precludes the gathering of such datasets via crowdsourcing, or via mixed initiatives where computers propose edges or nodes and humans accept or reject them. Finally, a clear next step for modeling more intelligent models is incorporating the kinds of relationships our metric captures into the model, either via constraints, different loss functions, or data augmentation techniques.

Chapter 6

CONCLUSION

In this dissertation, we presented various challenges that humans face when trying to understand and evaluate machine learning models, including the black box problem, the brittleness problem and the lack of metrics to evaluate higher level thinking. Our thesis is that model-agnostic explanations and evaluation enable efficient communication between machine learning models and humans, empowering humans to better evaluate, improve and assess trust in models. To this end, we presented two different kinds of model-agnostic techniques aimed at explaining individual predictions: LIME (Chapter 2) and Anchors (Chapter 3). In Chapter 4 we presented different kinds of explanations, aimed at uncovering brittleness bugs. SEAs and SEARs expose situations where models systematically change predictions in the presence of perturbations with no semantic significance. Finally, in Chapter 5 we presented preliminary work on Implication Consistency, a new kind of metric that allows humans to evaluate machine learning models as properties characteristic of higher level thinking, such as logical consistency. In this chapter, we generalize from our experiences by highlighting common themes across chapters, and by identifying open challenges and opportunities for future research.

6.1 Lessons Learned

Different information needs require different explanations

While it is easier to think of models as either "interpretable" or "black-box", or conversely to try to create one-size-fits-all explanations, this view misses the point of interpretability or explanations. Users have different goals, and an explanation that is perfectly suited to a goal (and thus renders a model "interpretable") may not be adequate for another goal,

even if the system is the same. For example, a fictitious credit system may wish to use LIME explanations to explain lending decisions to its end-users - this way, they can reason counterfactually about which actions they could take to improve their chances of getting a loan, by considering the weights in the explanation. An anchor explanation that gave them a sufficient condition for a loan denial would not serve this purpose, even though it may be very useful for the system designer trying to understand regions of the space where the prediction is more stable (e.g. users with extreme FICO scores). Using SEAs and SEARs on the end-user text applications may uncover bugs that would not be detected by either LIME or Anchor explanations, or even if the model being used is inherently "transparent" like a linear model (e.g. a user is more likely to get a loan approved if he describes his job as "Director of Marketing" rather than "Marketing Director"). Conversely, great care in the design and development of explanations must be placed in understanding who will use the explanations, how they will be used and the intended goal. Although harder to do than automated experiments, evaluation should also reflect this, and explanation or debugging systems should be evaluated as to whether they aid humans in specific tasks. We did this explicitly in this dissertation via a variety of user studies, and by showing qualitative examples of the kinds of discoveries that were made possible by explanations.

Understanding models brings human goals into focus

An academic paper is the only scenario where machine learning models act in a vacuum, with the single goal of making correct predictions. In real life, models are embedded into products or systems, and are deployed with hard-to-quantify goals such as "increase user satisfaction", or "help people make better decisions". In these situations, different mistakes have wildly different costs, and the correlation between accuracy and quality is not linear (e.g. a 10% gain in translation accuracy is much more significant for a 90% accurate system than a 10% one, as the latter is still not useful after the boost). Aggregate metrics (e.g. accuracy) are immensely useful, and models that are more accurate are typically better at the end-goals than less accurate ones. However, relying exclusively on aggregate measures often has the unfortunate

effect of making humans confuse the metric with the end goal. Trying to understand models inevitably acts as a counterbalance to this effect, as users are forced to think about the task and their goals as they contemplate and evaluate their new understanding of the model's behavior. In many cases, understanding leads users to realize that an aggregate measure is hiding a form of "reward hacking" - which may be great for the aggregate metric but terrible for the end goal. In other cases, users gather that models are behaving sensibly, and their trust in such models increases. Perhaps most often a mixture occurs: models are sensible but understanding them points out avenues of potential improvement. In any case, after seeing this scenario play out for a variety of state-of-the-art models for various domains and tasks, it is our belief that spending some time in trying to understand model behavior (defined broadly to include potential bugs and pitfalls) is a worthy investment in any real application of machine learning.

Humans have a lot to contribute

Machine learning technology is not at a point where we can discard human involvement, and it is hard to envision a time when it will reach that point. In addition to the fact that humans have a lot of background knowledge about various subjects, humans are able to cope with the ambiguity of real life end-goals that are hard to describe or quantify. Our experiments repeatedly bear witness to this fact: slight human involvement leads to systems that generalize much better in Chapter 2, or that are less prone to bugs in Chapter 4. Similarly, the insights provided by explanations in Chapters 2, 3 and 5 make further progress much more manageable by pointing us in specific directions for improvement. Finally, humans are the ones who (still) decide whether or not a model will be used, and helping them in that decision is an important goal.

6.2 Open Challenges and Opportunities for Future Research

We highlight here what we consider are important challenges and opportunities in the human - ML interaction. This list is by no means exhaustive, but contains what we believe are

important directions for the field.

Global understanding

We highlighted in Section 2.3 that there is a trade-off between model flexibility and interpretability: one cannot use a model whose behavior is very complex, yet expect humans to fully comprehend it globally. However, we have noted that interpretability is not a binary concept: we have shown in Chapters 2, 3, 4 that Submodular Pick and SEARs have a large impact in global understanding, leading to better human performance in various tasks when compared to random explanations or no explanations, even though our test subjects definitely did not fully comprehend the models used. While we have applied the paradigm of using carefully chosen explanations in the case of LIME and Anchors with success, we believe there is an unexplored opportunity in coming up with explanations that are global in nature. Just like the "language" of SEARs is different from SEAs (rules vs adversarial examples), a challenge for global understanding is coming up with a "language" of visual and textual artifacts to communicate global properties of models to the user, which may be different than the language used for individual explanations. While global understanding is particularly important for assessing trust in a model as a whole (before deployment), most of current research has been devoted to explaining individual predictions.

Understanding multiple models

Even though we have an experiment where the task is to pick the best out of two models in Section 2.7.2, most of our discussion has focused on understanding, debugging or evaluating a single model. As machine learning technology matures, being able to understand the differences between two or more models becomes crucial for version control, and for making deployment decisions. Furthermore, machine learning models usually come with an array of hyperparameters, and understanding the impact of such choices can lead to models that better conform to real life end-goals. Explaining each model separately as we did in 2.7.2 is obviously a starting point, but more sophisticated techniques that take into account regions of

the space where the models agree / disagree can surely be devised. Most importantly, similar to the case of global understanding, researchers need to develop a language of explanations for multiple models, which corresponds to the task and its complexities.

Better understanding of human perception of explanations

While we demonstrated the usefulness of explanations in a variety of tasks (model selection, feature engineering, debugging, predicting model behavior on new instances), we did not explicitly study how humans perceive explanations. It is clear from our experiment in Section 3.5.2 that different explanations result in different mental models, where human confidence and accuracy varies dramatically. User studies aimed at understanding how different ways of presenting information cause human confidence and accuracy to go up or down, and how users' mental models change as they are exposed to explanations would certainly better inform the design and communication of explanations, present or future. Such studies would also inform us as to whether it is worth exposing a measure of "explanation confidence", such as the goodness of fit of the local linear model that generates a LIME explanation, or if omitting explanations with low confidence altogether would result in better human mental models. They would also allow us to measure the effect of combining explanations - for example, instead of using LIME or Anchors, we can envision a system where both are used at different times. Finally, such studies would shed light both on the impact of potentially conflicting explanations such as the one in Figure 3.1b, which may be completely faithful to the model but a potential source of confusion.

Leveraging Human Feedback

There is a large body of research on providing feedback to models for "weakly supervised learning" - i.e. with very limited amounts of labeled data. The interaction usually involves having the user define constraints on the distributions of model predictions given certain features [15, 21], and / or labeling carefully chosen instances as in active learning [70]. A challenge virtually ignored by the literature is how to incorporate human feedback into a

classifier that has been trained on a reasonable amount of data (which is the most common case), where adding more labeled data usually has little impact unless the quantity is very large. This kind of feedback is required, however, when users discover that models are exhibiting behaviors that go against business rules (e.g. by being racist), or that are using reasoning that humans know is flawed. While we have shown great benefits of using the simplest forms of feedback on explanations (removing words in 2.7.3, augmenting training data in 4.5.4), we have barely began to explore the possibilities for human interaction and "guiding" of models. There are two main challenges involved in this research direction. First, designing expressive yet intuitive forms of feedback. It is not reasonable to expect users to label hundreds of examples in order to change a model's behavior, if more compact ways of communicating the change are viable. Feedback on explanations is a potential direction, as explanations are already trying to distill the model's reasoning, although specifying what the user interaction would be remains a challenge. Second, incorporating such feedback in a model-agnostic way is a very hard technical problem, as even designing specific models that are able to use human background knowledge remains a research challenge. As these challenges are tackled by the research community, a true "human-in-the loop" approach can be envisioned, where the model explains itself to the user in an interactive way, gets explicit and implicit feedback in order to improve and to explain itself better, and repeats the process.

6.3 Conclusion

This dissertation addressed the fundamental communication problem between humans and machine learning models. As machine learning becomes more embedded into everyday life, such communication is not only beneficial in that it results in better models, but also becomes a necessity for legal and moral reasons. To this end, we proposed model-agnostic explanations and evaluation, and demonstrated that these empower humans to better understand, evaluate, improve and assess trust in models. Together, these contributions provide a set of tools that is invaluable for understanding, debugging and evaluating machine learning models, as well as laying the foundation for a research area that is crucial for the maturing of the field.

BIBLIOGRAPHY

[1] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Human Factors in Computing Systems (CHI)*, 2015.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*, 2015.

[4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11, 2010.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Aayush Bansal, Ali Farhadi, and Devi Parikh. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision (ECCV)*, 2014.

[7] Aayush Bansal, Ali Farhadi, and Devi Parikh. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision (ECCV)*, 2014.

[8] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-

boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics (ACL)*, 2007.

[9] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, January 2015.

[10] J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. MIT, 2009.

[11] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Knowledge Discovery and Data Mining (KDD)*, 2015.

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

[13] Mark W Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, pages 24–30, 1996.

[14] Luc De Raedt and Kristian Kersting. Probabilistic inductive logic programming. chapter Probabilistic Inductive Logic Programming, pages 1–27. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 3-540-78651-1, 978-3-540-78651-1. URL `http://dl.acm.org/citation.cfm?id=1793956.1793958`.

[15] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM, 2008.

[16] Mary T. Dzindolet, Scott A. Peterson, Regina A. Pomranky, Linda G. Pierce, and Hall P. Beck. The role of trust in automation reliance. *Int. J. Hum.-Comput. Stud.*, 58 (6), 2003.

[17] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Deijing Dou. HotFlip: White-Box Adversarial Examples for NLP. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[18] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[19] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4), July 1998.

[20] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014. ISSN 1931-0145.

[21] Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul): 2001–2049, 2010.

[22] Matt A Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

[23] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

[24] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, volume 1, page 3, 2017.

[25] Alex Groce, Todd Kulesza, Chaoqiang Zhang, Shalini Shamasunder, Margaret Burnett, Weng-Keen Wong, Simone Stumpf, Shubhomoy Das, Amber Shinsel, Forrest Bice, and Kevin McIntosh. You are the only possible oracle: Effective test selection for end users of interactive machine learning systems. *IEEE Trans. Softw. Eng.*, 40(3), 2014.

[26] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 192–202, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[28] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 690–696, 2013.

[29] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Conference on Computer Supported Cooperative Work (CSCW)*, 2000.

[30] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[31] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics (NAACL)*, 2018.

[32] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[33] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[34] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[35] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[36] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. Leakage in data mining: Formulation, detection, and avoidance. In *Knowledge Discovery and Data Mining (KDD)*, 2011.

[37] Emilie Kaufmann and Shivaram Kalyanakrishnan. Information complexity in bandit subset selection. In *Proceedings of the Twenty-sixth nnual Conference on Learning Theory (COLT 2013)*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 228–251. JMLR, 2013.

[38] Vahid Kazemi and Ali Elqursh. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv:1704.03162*, 2017.

[39] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[40] Been Kim, Cynthia Rudin, and Julie A Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1952–1960. Curran Associates, Inc., 2014.

[41] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[42] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

[43] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[44] Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Knowledge Discovery and Data Mining (KDD)*, 2015.

[45] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

[46] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. February 2014.

[47] Josua Krause, Adam Perer, and Kenney Ng. Interacting with predictions: Visual inspection of black-box machine learning models. 2016.

[48] Todd Kulesza, Simone Stumpf, Weng-Keen Wong, Margaret M. Burnett, Stephen Perona, Andrew Jensen Ko, and Ian Oberst. Why-oriented end-user debugging of naive bayes text classification. *TiiS*, 1:2:1–2:31, 2011.

[49] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Intelligent User Interfaces (IUI)*, 2015.

[50] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1675–1684, New York, NY, USA, 2016. ACM. doi: 10.1145/2939672.2939874.

[51] Mirella Lapata, Rico Sennrich, and Jonathan Mallinson. Paraphrasing revisited with neural machine translation. In *European Chapter of the ACL (EACL)*, 2017.

[52] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*, 2014.

[53] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 2015.

[54] Jiwei Li, Will Monroe, and Daniel Jurafsky. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016.

[55] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2119–2128, New York, NY, USA, 2009. ACM. doi: 10.1145/1518701.1519023.

[56] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[57] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[58] David Martens and Foster Provost. Explaining data-driven document classifications. *MIS Q.*, 38(1), 2014.

[59] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems (NIPS)*. 2013.

[60] George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information, 1956.

[61] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38, 1995.

[62] Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*, 2017.

[63] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529, 2015.

[64] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[65] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

[66] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. Investigating statistical machine learning as a tool for software development. In *Human Factors in Computing Systems (CHI)*, 2008.

[67] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. Investigating statistical machine learning as a tool for software development. In *Human Factors in Computing Systems (CHI)*, 2008.

[68] Kayur Patel, Naomi Bancroft, Steven M. Drucker, James Fogarty, Andrew J. Ko, and James Landay. Gestalt: Integrated support for implementation and analysis in machine learning. In *User Interface Software and Technology (UIST)*, 2010.

[69] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[70] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *J. Mach. Learn. Res.*, 7:1655–1686, December 2006. ISSN 1532-4435.

[71] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[72] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-2124`.

[73] Mengye Ren, Ryan Kiros, and Richard S. Zemel. Exploring models and data for image question answering. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 2953–2961, Cambridge, MA, USA, 2015. MIT Press.

[74] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Knowledge Discovery and Data Mining (KDD)*, 2016.

[75] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. In *Human Interpretability in Machine Learning workshop*, ICML '16, 2016.

[76] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

[77] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

856–865. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-1079`.

[78] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *CoRR*, abs/1707.02812, 2017.

[79] Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, and Sameer Singh. Towards extracting faithful and descriptive representations of latent variable models. In *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*, 2015.

[80] Peter Schmidt and Ann D Witte. *Predicting Recidivism in North Carolina, 1978 and 1980*. Inter-university Consortium for Political and Social Research, 1988.

[81] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, and Jean-Franç Crespo. Hidden technical debt in machine learning systems. In *Neural Information Processing Systems (NIPS)*. 2015.

[82] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2017.

[83] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[84] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11, 2010.

[85] Simone Stumpf, Vidya Rajaram, Lida Li, Margaret Burnett, Thomas Dietterich, Erin Sullivan, Russell Drummond, and Jonathan Herlocker. Toward harnessing user

feedback for machine learning. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, pages 82–91, New York, NY, USA, 2007. ACM. doi: 10.1145/1216295.1216316.

[86] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[87] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[88] JÃűrg Tiedemann. Parallel data, tools and interfaces in opus. In *International Conference on Language Resources and Evaluation (LREC)*, 2012.

[89] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2006.

[90] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 2015.

[91] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, pages 705–718. Springer, 2008.

[92] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial Intelligence and Statistics (AISTATS)*, 2015.

[93] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. Or's of and's for interpretable classification, with application to context-aware recommender systems. *arXiv:1504.07614*, 2015.

[94] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as

possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, 2017.

[95] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[96] John Wieting and Kevin Gimpel. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[97] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2015.

[98] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, 2015.

[99] Peng Zhang, Jiuling Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[100] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[101] Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. Training set debugging using trusted items. In *AAAI Conference on Artificial Intelligence*, 2018.

[102] Yan Zhang, Jonathon Hare, and Adam PrÃijgel-Bennett. Learning to count objects in natural images for visual question answering. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B12Js_yRb`.

[103] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[104] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.

[105] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7W: Grounded Question Answering in Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.