An Analysis of Neural Rationale Models and Influence Functions for Interpretable Machine Learning

by

Yiming Zheng

S.B., Computer Science and Engineering, Massachusetts Institute of Technology (2023)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Authored by:	Yiming Zheng Department of Electrical Engineering and Computer Science May 12, 2023
Certified by:	Julie A. Shah H.N. Slater Professor of Aeronautics and Astronautics Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

An Analysis of Rationale Models and Influence Functions for Interpretable Machine Learning

by

Yiming Zheng

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In recent years, increasingly powerful machine learning models have shown remarkable performance on a wide variety of tasks and thus their use is becoming more and more prevalent, including deployment in high stakes settings such as for medical and legal applications. Because these models are complex, their decision process is hard to understand, suggesting a need for model interpretability. Interpretability can be deceptively challenging. First, explanations for a model's decision on example inputs may appear understandable. However, if the underlying *explanation method* is not interpretable, more care must be taken before making a claim about the interpretability of the explanation method. Second, it can be difficult to use interpretability techniques efficiently on large models with many parameters.

Through the lens of the first challenge, we examine *neural rationale models*, which are popular for interpretable predictions of natural language processing (NLP) tasks. In these, a selector extracts segments of the input text, called *rationales*, and passes these segments to a classifier for prediction. Since the rationale is the only information accessible to the classifier, it is plausibly *defined* to be the explanation. However, through both philosophical perspectives and empirical studies, we argue rationale models may be less interpretable than expected. We call for more rigorous evaluations of these models to ensure desired properties of interpretability are indeed achieved. Through the lens of the second challenge, we study *influence functions* which explain a model's output by tracing the model decision process back to the training data. Given a test point, influence functions compute an influence score for each training point representing how influential it is on the model's decision with the test point as input. While expensive to compute on large models with many parameters, we aim to gain intuition on influence functions in low dimensional settings and develop simple, cheap to compute heuristics which are competitive with influence functions.

Thesis Supervisor: Julie Shah
Title: H.N. Slater Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank Serena Booth and Yilun Zhou for their extensive support and mentorship throughout my undergraduate research projects as well as my work for this thesis. I would also like to thank my advisor Prof. Julie Shah. Finally, I would like to thank all of my friends and family for their constant support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

While complex machine learning models such as deep neural networks achieve exceptional performance on a wide variety of tasks, as these models are being deployed in high stakes settings, understanding the prediction process for these models becomes more and more important. Techniques in interpretable machine learning aim to make the model outputs understandable to humans, giving not just the output, but also *why* the model chose the output that it did on a certain input. As many complex, high performing machine learning models are notorious for picking up signals in the form of complicated correlations, it is necessary to understand how the model chose its output to make fair and ethical decisions. For example, consider a model forecasting crime rates in different neighbors to help decide how to optimally allocate police officers. Even if the model performs with a high accuracy, interpretability is necessary for detecting if racial biases are present in the training data and whether the model has learned to make predictions based on these biases (Lipton, 2018). Additionally, model interpretability can be important for the human user of the model. It can help the user understand why the model gave a certain output by, for example, highlighting important regions of the input, or important training examples, which were considered most in the process of determining the output. Consider a model which takes as input a medical scan of a patient and outputs a diagnosis. If a doctor using this model sees an input where the model predicts the patient has a certain disease, but the highlighted regions of the image look very different from what is typical in a patient with the disease, the doctor now knows to proceed with extra caution before using the model's diagnosis (Ahmad et al., 2018).

There are two main approaches for explaining a model's output on a given input: feature attribution methods and training data based explanation. For a given input, feature attribution methods work by highlighting a subset of features of that input which are most important to the model's output on that input, for some definition of important. On the other hand, training based explanation methods specify which training instances are most influential on the model's prediction for a given input. Both feature attribution and training based explanation are examples of *post-hoc* interpretability methods, which provide explanation after the model of interest is trained (Molnar, 2022).

While model interpretability is important in many practical settings, it is deceptively hard to achieve. First off, a model explanation method might be misleadingly deemed interpretable if the explanation method's outputs appear to be understandable to a human, but the underlying explanation method isn't in fact interpretable. Examples of such an explanation method would include if the method is either not faithful to the model and training data, or if the method is entirely a black box process itself. In fact, while various saliency based interpretability methods (which are a type of feature attribution method) appear to highlight important features in the input, they have been shown to be independent of both the data and the model and thus misleading and not truly faithful to the model's decision making process (Adebayo et al., 2018). Thus, interpretability needs to be approached with care. A second challenge is that large and complex models with many parameters which are frequently used in practice, such as deep neural networks, inherently have complicated decision making processes. Thus, it can be difficult to provide an interpretable explanation for this process in a tractable manner.

In this thesis, we examine the two challenges mentioned above through studies of two methods in interpretable machine learning. We begin by analyzing neural rationale models and investigate how the challenge of ensuring the underlying explanation method is interpretable applies. Neural rationale models use a similar idea to feature attribution, but instead of post-hoc interpretability, they aim to be "inherently interpretable" via a two part architecture. First, a selector model extracts segments of the input text, called *rationales*. The selector model then passes these rationales to a classifier model which use these rationales for prediction. Since the rationale is the only information accessible to the classifier, it is deemed the explanation. Is such a characterization unconditionally correct? We argue to the contrary, with both philosophical perspectives and empirical evidence suggesting that rationale models are, perhaps, less rational and interpretable than expected. To ensure desired properties of interpretability are achieved, more rigorous evaluations of these models are required.

Next, we study influence functions through the lens of the second challenge mentioned above: making model interpretability techniques more efficient to compute. To this end, we aim to gain intuition on influence functions, which is one type of training based explanation. The influence score of a training point with respect to a test point is defined as how much the model's prediction confidence on that test point changes when the training point is removed from the training set and the model is retrained. The training points with the smallest (most negative) influence scores are said to have the most influence on the prediction confidence of the test point. The theory of influence functions make assumptions which may not necessarily hold in practice. Influence functions also use several approximations for not only computing influence, but also approximating the influence of a group of training points through the combination of the influences of the individual points in the group. Finally, influence functions are expensive to compute in practice. Thus, we aim to gain intuition on how influence functions behave in low dimensional settings on logistic regression models and develop cheap heuristic algorithms which perform competitively with influence functions. We leave to future work extending the intuition and heuristics to more complex models and higher dimensional data.

Chapter 2

Neural Rationale Models

Neural rationale models are a popular for interpretable predictions of NLP tasks. They are made up of a two part framework. A selector model extracts segments of the input text, called rationales, which it passes to a classifier for prediction. The selector and classifier models are typically both implemented as neural networks and trained jointly. While the rationale is deemed the explanation and may be a humanly understandable subset of the input text (e.g. all the positive phrases in the input on a positive prediction in a sentiment analysis task), the underlying explanation method is via the selector model which is a neural network (and thus still a black box). Before we discuss neural rationale models in more detail, we first motivate their potential utility by discussing post-hoc interpretability.

2.1 Post-hoc Interpretability

Most interpretability efforts focus on *post-hoc* interpretation. For a specific input, these methods generate an explanation by analyzing model behaviors such as gradient (Simonyan et al., 2013; Sundararajan et al., 2017; Smilkov et al., 2017; Selvaraju et al., 2017; Shrikumar et al., 2017; Chattopadhay et al., 2018; Bykov et al., 2022) or prediction on perturbed (Ribeiro et al., 2016; Lundberg and Lee, 2017; Chang et al., 2018; Covert et al., 2021) or reduced (Feng et al., 2018) inputs. However, evaluations of these methods highlight various problems. For example, Adebayo et al. (2018)

showed that many methods can generate seemingly reasonable explanations even for random neural networks. Kindermans et al. (2019) show several of these methods are not reliable when a transformation on the input data with no effect on the model, such as a constant shift, is introduced. Zhou et al. (2022a) found that many methods fail to identify features known to be used by the model. Zhou et al. (2022b) share the same principles as us, but also focus on general post-hoc interpretations of arbitrary black-box models, while we focus on neural rationale models.

2.2 Neural Rationale Models

With no resolution in sight for explaining black box models, *inherently interpretable* models, which self-explain while making decisions, are often favored. Neural rationale models, shown in Figure 2-1 (top), are the most popular in NLP (Lei et al., 2016; Bastings et al., 2019; Yu et al., 2019; Jain et al., 2020): in them, a selector processes the input text, extracts segments (i.e. *rationale*) from it, and sends *only* the rationale to the predictor. Since the rationale is the only information accessible to the predictor, it arguably serves as the *explanation* for the prediction.

While the bottleneck structure in rationale models defines a causal relationship between rationale and prediction, we caution against equating this structure with inherent interpretability without additional constraints. Notably, if both the selector and the classifier are sufficiently flexible function approximators (e.g. neural networks), the bottleneck structure provides *no* intrinsic interpretability as the selector and classifier may exploit imperceptible messages, as shown in Figure 2-1 (bottom).

We perform a suite of empirical analyses to demonstrate how rationales lack interpretability. Specifically, we present modes of instability of the rationale selection process under minimal and meaning-preserving sentence perturbations on the Stanford Sentiment Treebank (SST, Socher et al., 2013) dataset. Through a user study, we further show that this instability is poorly understood by people—even those with advanced machine learning knowledge. We find that the exact form of interpretability induced by neural rationale models, if any, is not clear. As a community, we

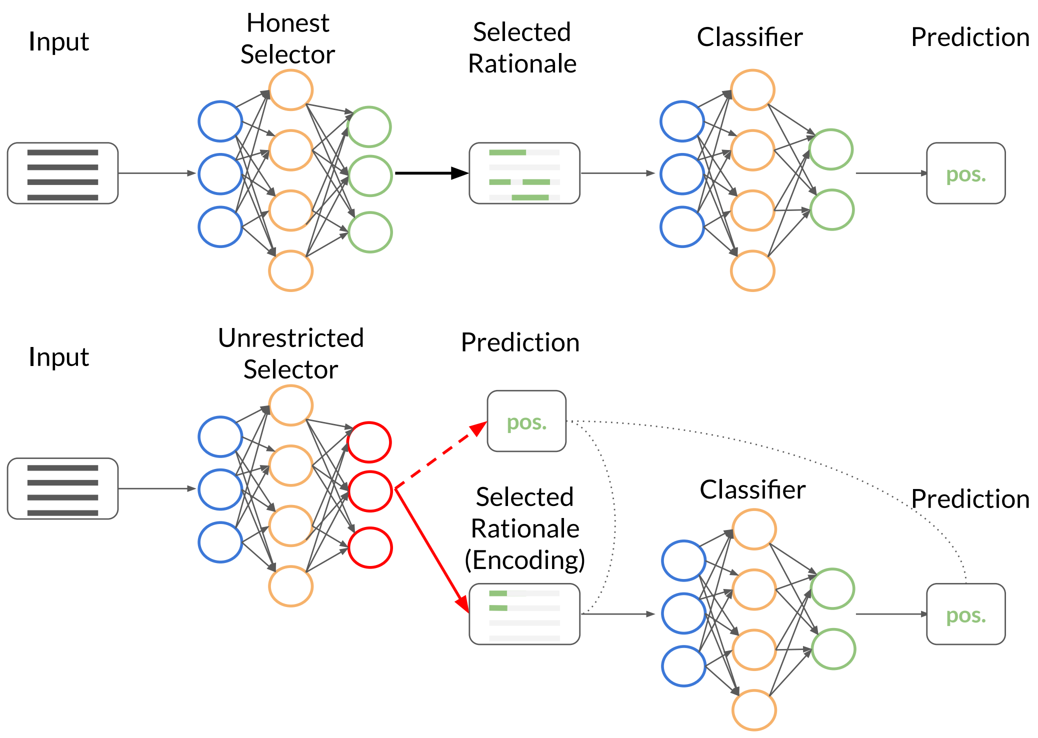

Figure 2-1: Top: an honest neural rationale model. We seek to understand the selector's process (the bold arrow), which should select words and phrases as rationale in an unbiased way, leaving the prediction to the classifier which receives this rationale. Bottom: a failure case of neural rationale models. As discussed in Section 2.4, an unrestricted selector may be able to make its own (relatively accurate) prediction, and "pass" it to the classifier via encoding it in the selected rationale.

must critically reflect on the interpretability of these models, and perform rigorous evaluations about any and all claims of interpretability going forward.

2.3 Related Work

Neural rationale models are largely deemed *inherently interpretable* and thus do not require *post-hoc* analysis. At a high level, a model has a selector and a classifier. For an input sentence, the selector first calculates the *rationale* as excerpts of the input, and then the classifier makes a prediction from *only* the rationale. Thus, the rationale is often *defined* as the explanation due to this bottleneck structure. The non-differentiable rationale selection prompts people to train the selector using policy gradient (Lei et al., 2016; Yu et al., 2019) or continuous relaxation (Bastings et al., 2019), or directly use a pre-trained one (Jain et al., 2020).

While rationale models have mostly been subject to less scrutiny, some evaluations have been carried out. Yu et al. (2019) proposed the notions of comprehensiveness and sufficiency for rationales, advocated as standard evaluations in the ERASER (DeYoung et al., 2019) dataset. Zhou et al. (2022a) noted that training difficulty, especially due to policy gradient, leads to selection of words known to not influence the label in the data generative model. Complementing these evaluations and criticisms, we argue from additional angles to be wary of interpretability claims for rationale models, and present experiments showing issues with existing models.

Most related to our work, Jacovi and Goldberg (2020) mention a Trojan explanation and dominant selector as two failure modes of rationale models. We pinpoint the same root cause of a non-understandable selector in Section 2.4. However, they favor rationales generated *after* the prediction, while we will argue for rationales being generated *prior* to the prediction. Also, in their discussion of contrastive explanations, their proposed procedure runs the model on out-of-distribution data (sentence with some tokens masked), potentially leading to arbitrary predictions due to extrapolation, a criticism also argued by Hooker et al. (2018).

2.4 Philosophical Perspectives

In neural rationale models, the classifier prediction causally results from the selector rationale, but does this property automatically equate rationale with explanation? We first present a "failure case." For a binary sentiment classification, we first train a (non-interpretable) classifier c' that predicts on the whole input. Then we *define* a selector s' that selects the first word of the input if the prediction is positive, or the first two words if the prediction is negative. Finally, we train a classifier c to imitate the prediction of c' but from the rationale. The $c' \to s' \to c$ model should achieve best achievable accuracy, since the actual prediction is made by the unrestricted classifier c' with full input access. Can we consider the rationale as explanation? No, because the rationale selection depends on, and is as (non-)interpretable as, the black-box c'. This failure case is shown in Figure 2-1 (bottom). Recently proposed introspective training (Yu et al., 2019) could not solve this problem either, as the selector can simply output the comprehensive rationale along with the original cue of first one or two words, with only the latter used by the classifier[1]. In general, a sufficiently powerful selector can make the prediction at selection time, and then pass this prediction via some encoding in the selected rationale for the classifier to use.

To hide the "bug," consider now s' selecting the three most positive or negative words in the sentence according to the c' prediction (as measured by embedding distance to a list of pre-defined positive/negative words). This model would seem very reasonable to a human, yet it is non-interpretable for the same reason. To recover a "native" neural model, we could train a selector s to imitate $c' \rightarrow s'$ via teacher-student distillation (Hinton et al., 2015), and the innocent-looking $s \rightarrow c$ rationale model remains equally non-interpretable.

Even without the explicit multi-stage supervision above, a sufficiently flexible selector s (e.g. a neural network) can implicitly learn the $c' \rightarrow s'$ model and essentially control the learning of the classifier c, in which case the bottleneck of succinct rationale affords no benefits of interpretability. So why does interpretability get lost (or fail to emerge)? The issue arises from not understanding the *rationale selection process*, i.e. selector s. If it is well-understood, we could determine its true logic to be $c' \rightarrow s'$ and reject it. Conversely, if we cannot understand *why* a particular rationale is selected, then accepting it (and the resulting prediction) at face value is not really any different from accepting an end-to-end prediction at face value.

In addition, the selector-classifier decomposition suggests that the selector should be an "unbiased evidence collector", i.e. scanning through the input and highlighting all relevant information, while the classifier should deliberate on the evidence for each class and make the decision. Verifying this role of the selector would again require its interpretability.

Finally, considering the rationale model as a whole, we could also argue that the rationale selector *should* be interpretable. It is already accepted that the classifier

[1]In fact, the extended rationale *helps* disguise the problem by appearing as much more reasonable.

can remain a black-box. If the selector is also not interpretable, then exactly what about the model is interpretable?

Architecturally, we can draw an analogy between the rationale in rationale models and the embedding representation in a typical end-to-end classifier produced at the penultimate layer. A rationale is a condensed feature extracted by the selector and used by the classifier, while, for example in image models, the image embedding is the semantic feature produced by the feature extractor and used by the final layer of linear classifier. Furthermore, both of them exhibit some interpretable properties: rationales represent the "essence" of the input, while the image embedding space also seems semantically organized (e.g. Figure 2-2 showing ImageNet images organized in the embedding space). However, this embedding space is rarely considered on its own as the explanation for a prediction, exactly because the feature extractor is a black-box. Similarly, the rationales by default should not qualify as the explanation either, despite its textual nature.

Finally, from a practical perspective, explanations should help humans understand the model's input-output behavior. Such a purpose is fulfilled when the human understands not only why an explanation leads to an output, but also *how* the explanation is generated from the input in the first place. Our emphasis on understanding the rationale selection process fulfills the latter requirement. Such a perspective is also echoed by Pruthi et al. (2020a), who argued that the practical utility of explanations depends crucially on human's capability of understanding how they are generated.

2.5 Empirical Investigation

As discussed above, truly interpretable rationale models require an understanding of the rationale selection process. However, since the selector is a sequence-to-sequence model, for which there is no standard methods for interpretability, we focus on a "necessary condition" setup of understanding the input-output behavior of the model in our empirical investigation. Specifically, we investigate rationale selection changes in response to meaning-preserving non-adversarial perturbation of individual words

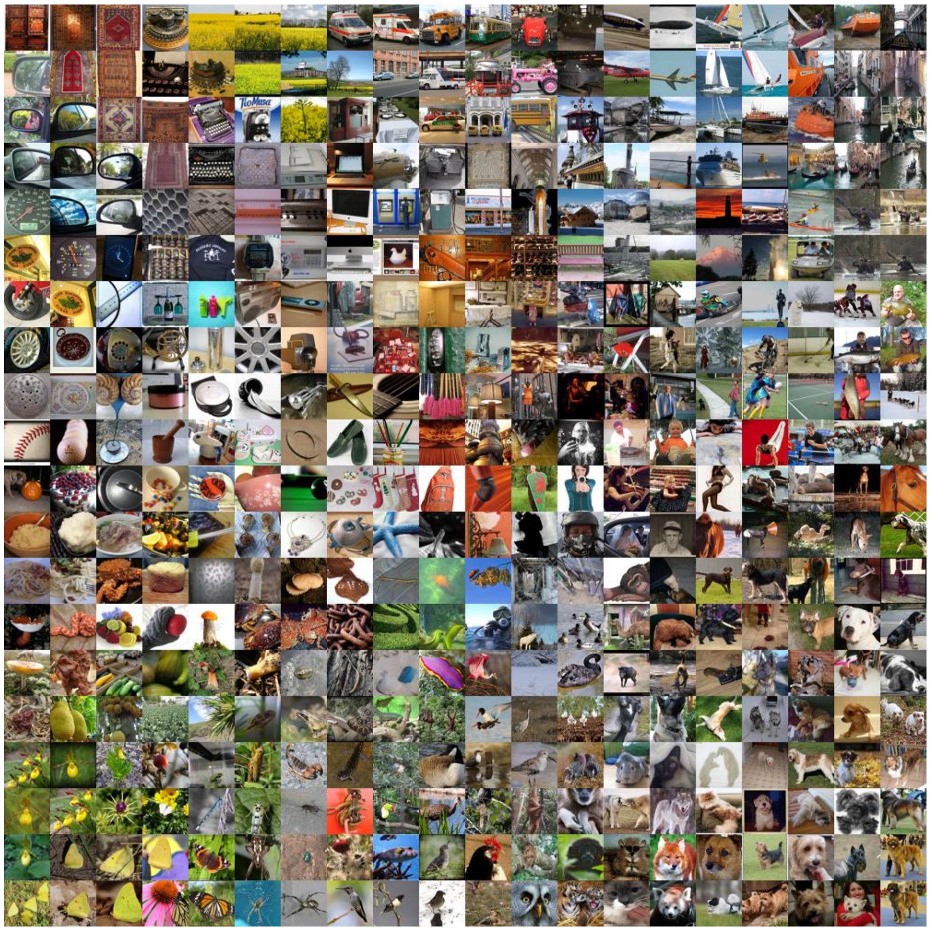

Figure 2-2: Embedding space visualization of an ImageNet classifier. Image from `https://cs.stanford.edu/people/karpathy/cnnembed/`.

in the input sentence.

2.5.1 Setup

On the 5-way SST dataset (Socher et al., 2013), we trained two rationale models, a continuous relaxation (CR) model (Bastings et al., 2019) and a policy gradient (PG) model (Lei et al., 2016). The PG model directly generates binary (i.e. hard) rationale selection. The CR model uses a $[0, 1]$ continuous value to represent selection and scales the word embedding by this value. Thus, we consider a word being selected as rationale if this value is non-zero. Our CR model achieves 47.3% test accuracy with 24.9% rationale selection rate (i.e. percentage of words in the input selected as rationale), and PG model 43.3% test accuracy with 23.1% rationale selection rate, consistent with those obtained by Bastings et al. (2019, Figure 4). Additional details are in Appendix A.1.

2.5.2 Sentence Perturbation Procedure

The perturbation procedure changes a noun, verb, or adjective as parsed by NLTK[2] (Loper and Bird, 2002) with two requirements. First, the new sentence should be natural (e.g., "I *observed* a movie" is not). Second, its meaning should not change (e.g. adjectives should not be replaced by antonyms).

For the first requirement, we `[MASK]` the candidate word and use the pre-trained BERT (Devlin et al., 2019) to propose 30 new choices. For the second requirement, we compute the union of words in the WordNet synset associated with each definition of the candidate words (Fellbaum, 1998). If the two sets share no common words, we mark the candidate invalid. Otherwise, we choose the top BERT-predicted word as the replacement.

We run this procedure on the SST test set, and construct the perturbed dataset from all valid replacements of each sentence. Table 2.1 lists some example perturbations (more in Appendix A.2). Table 2.2 shows the label prediction distribution on the original test set along with changes due to perturbation in parentheses, and confirms that the change is overall very small. Finally, a human evaluation checks the perturbation quality, detailed in Appendix A.3. For 100 perturbations, 91 were rated to have the same sentiment value. Furthermore, on all 91 sentences, the same rationale is considered adequate to support the prediction after perturbation as well.

A pleasurably jacked-up **piece**/*slice* of action moviemaking .
The **use**/*usage* of CGI and digital ink-and-paint make the thing look really slick .

Table 2.1: Sentence perturbation examples, with the original word in **bold** replaced by the word in *italics*.

[2]i.e. NN, NNS, VB, VBG, VBD, VBN, VBP, VBZ, and JJ

	0	1	2	3	4
CR	8.0 (-0.6)	41.5 (+1.5)	8.9 (-0.4)	28.6 (+1.0)	13.0 (-1.5)
PG	8.6 (-1.6)	40.7 (-1.3)	1.6 (-0.2)	33.9 (+5.0)	15.2 (-1.9)

Table 2.2: The percentage of predicted labels on the original test set, as well as the differences to that on the perturbation sentences in parentheses.

2.5.3 Results

Now we study the effects of perturbation on rationale selection change (i.e. an originally selected word getting unselected or vice versa). We use only perturbations that maintain the model prediction, as in this case, the model is expected to use the same rationales according to human evaluation.

Qualitative Examples Table 2.3 shows examples of rationale changes under perturbation (more in Appendix A.4). Indeed, minor changes can induce nontrivial rationale change, sometimes far away from the perturbation location. Moreover, there is no clear relationship between the words with selection change and the perturbed word.

PG	The **story**/*narrative* loses its bite in a last-minute happy ending that 's even less plausible than the rest of the picture .
PG	A pleasant ramble through the sort of idoosyncratic terrain that Errol Morris **has**/*have* often dealt with ... it does possess a loose , lackadaisical charm .
CR	I love the way that it took chances and really asks you to take these **great**/*big* leaps of faith and pays off .
CR	Legendary Irish **writer**/*author* Brendan Behan 's memoir , Borstal Boy , has been given a loving screen transferral .

Table 2.3: Rationale change example. Words selected in the original only, perturbed only, and both are shown in red, blue, and green, respectively.

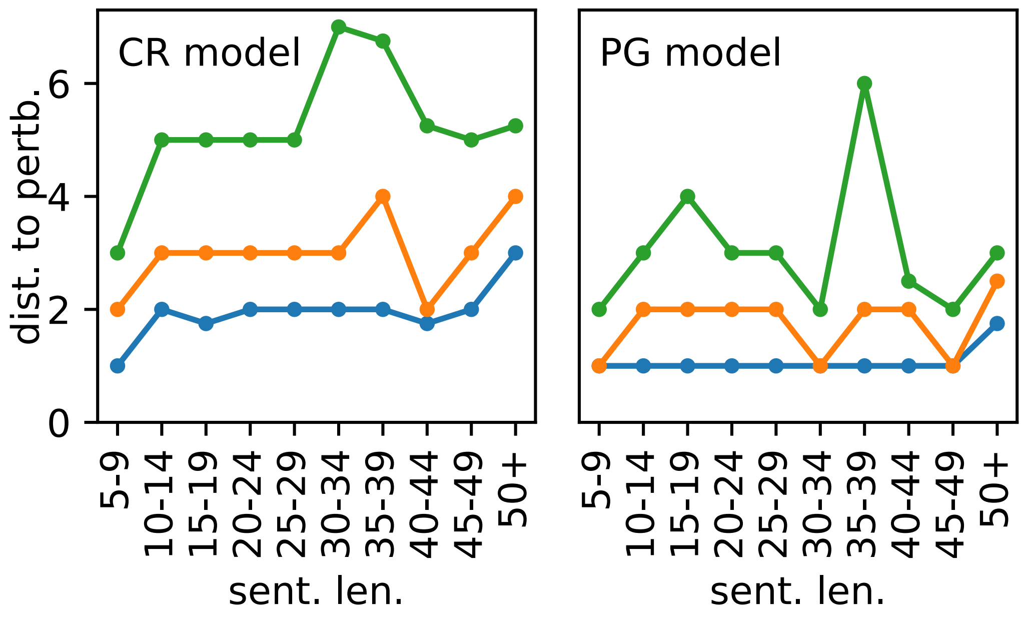

Figure 2-3: Scatter plots showing three quartiles of distance between indirect rationale change to perturbation, grouped by sentence length.

Rationale Change Freq. Quantitatively, we first study how often rationales change. Table 2.4 shows the count frequency of selection changes. Around 30% (non-adversarial) perturbations result in rationale change (i.e. non-zero number of changes). Despite better accuracy, the CR model is less stable and calls for more investigation into its selector.

# Change	0	1	2	3	4	≥ 5
CR	66.5%	25.5%	6.8%	1.0%	0.1%	0.1%
PG	77.4%	21.4%	1.1%	0.1%	0%	0%

Table 2.4: Frequency of number of selection changes.

Locations of Selection Change Where do these changes occur? 29.6% and 78.3% of them happen at the perturbed word for the CR and PG models respectively. For the CR model, over 70% of rationale changes are due to replacements of *other* words; this statistic is especially alarming. For these indirect changes, Figure 2-3 shows the quartiles of distances to the perturbation for varying sentence lengths. They are relatively constant throughout, suggesting that the selection uses mostly local

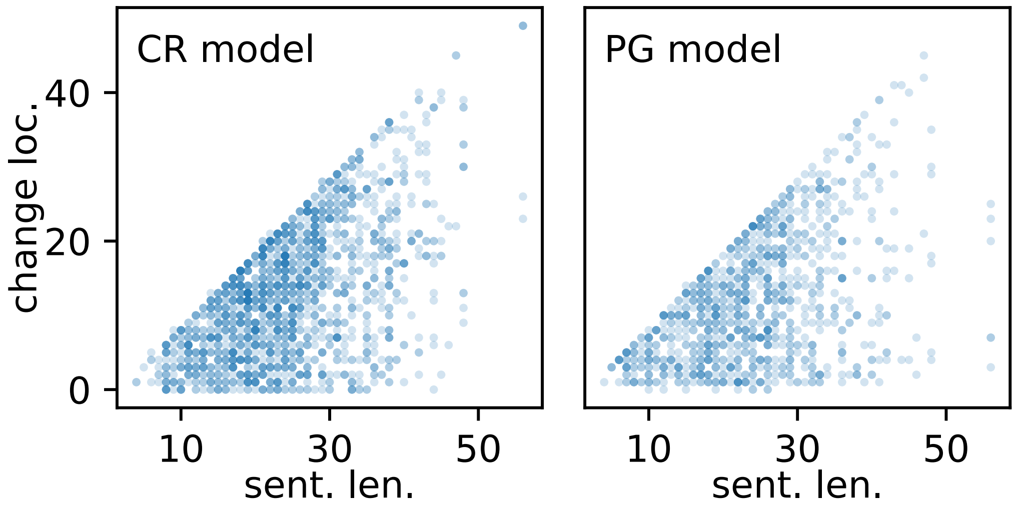

Figure 2-4: Locations of all selection changes, with each one shown as a dot.

information. However, the "locality size" for CR is about twice as large, and changes often occur five or more words away from the perturbation.

We also compute the (absolute) location of the rationale changes, as plotted in Figure 2-4, where each dot represents an instance. The rationale changes are distributed pretty evenly in the sentence, making it hard to associate particular perturbation properties to the resulting selection change location.

Sentence-Level Stability Are all the rationale changes concentrated on a few sentences for which every perturbation is likely to result in a change, or are they spread out across many sentences? We measure the *stability* of a sentence by the number of perturbations inducing rationale changes. Obviously, a sentence with more valid perturbations is likely to also have more change-inducing ones, so we plot the frequency of sentences with a certain stability value separately for different total numbers of perturbations in Figure 2-5. There are very few highly unstable sentences, suggesting that the selection change is a common phenomenon to most of the sentences, further adding to the difficulty of a comprehensive understanding of the selector.

Part of Speech Analysis Our final automated analysis studies the part-of-speech (POS) composition of selection changes. As Table 2.5 shows, adjectives and adverbs

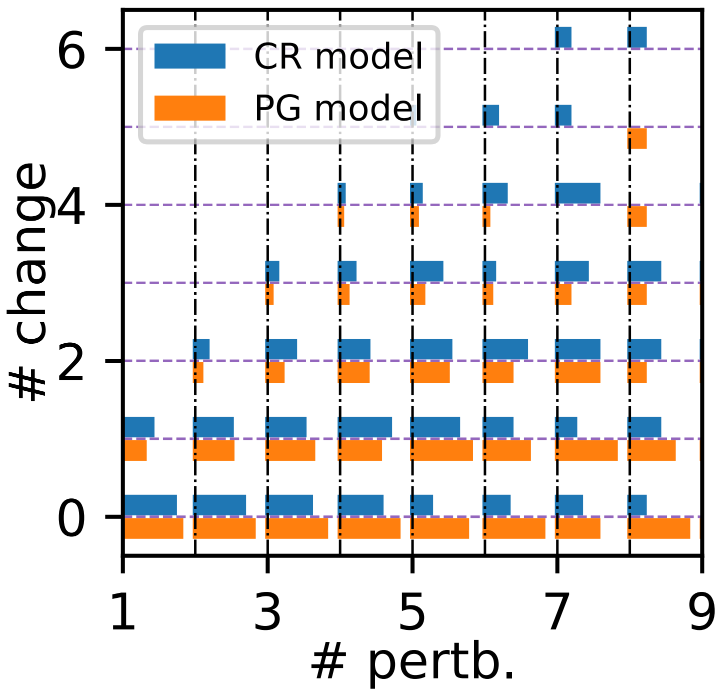

Figure 2-5: For sentences with a certain number of valid perturbations, the corresponding column of bar chart shows the count frequency of perturbations that result in any rationale change.

POS (frequency)	noun (19.2%)	verb (14.3%)	adj. (10.1%)	adv. (5.8%)	proper n. (4.4%)	pron. (4.9%)	other (41.3%)
CR change / all	37.1% / 34.3%	21.9% / 16.0%	14.2% / 24.8%	8.9% / 11.3%	3.5% / 5.8%	2.5% / 1.0%	11.9% / 6.8%
PG change / all	42.7% / 33.6%	30.2% / 16.6%	20.6% / 30.6%	2.4% / 12.9%	1.8% / 3.4%	0.4% / 0.5%	1.9% / 2.4%

Table 2.5: Part of speech (POS) statistics. The top row shows the POS composition of the test set sentences. The bottom two rows show POS composition for changed rationale words and for all rationale words.

are relatively stable, as expected because they encode most sentiments. By contrast, nouns and verbs are less stable, probably because they typically represent factual "content" that is less important for prediction. The CR model is especially unstable for other POS types such as determiner and preposition. Overall, the instability adds to the selector complexity and could even function as subtle "cues" described in Section 2.4.

User Study on Selector Understanding While the automated analyses reveal potential obstacles to selector understanding, ultimately the problem is the lack of understanding by users. The most popular way to understand a model is via input-output examples (Ribeiro et al., 2020; Booth et al., 2021), and we conduct a user study in which we ask participants (grad students with ML knowledge) to match rationale patterns with sentences before and after perturbation on 20 instances, after observing 10 true model decisions (details in Appendix A.5). Unsurprisingly, participants get 45 correct out of 80 pairs, basically at the random guess level, even as some participants use reasons related to grammar and atypical word usage (which are apparently ineffective), along with "lots of guessing". This result confirms the lack of selector understanding even under minimal perturbation, indicating more severity for completely novel inputs.

Chapter 3

Influence Functions

Influence functions are a type of explanation method for arbitrary models which are trained with data. They provide insight into the model's prediction process on a specified test instance by tracing the prediction back to the most influential training instances. While the theory of influence functions is mathematically sound under certain conditions, several of these conditions do not typically hold in practical settings. Additionally, to make the computation of influence functions run fast enough to be useful in practice, several approximations must be used, potentially compromising the accuracy of these computations. Even with these approximations, computing influence functions is still quite expensive.

Influence functions have been used in the past for not only model understanding, but a variety of other tasks such as training a model which is robust to adversarial inputs, debugging domain mismatches, and fixing mislabeled data (Koh and Liang, 2017), however they have since been shown to be unstable and fragile (K and Søgaard, 2021; Basu et al., 2020). Additionally, while influence functions have been used as a means of model understanding on large models with many parameters, such as natural language processing tasks (Han et al., 2020; Zhang et al., 2021a; Zylberajch et al., 2021), their effectiveness has been questioned (Kocijan and Bowman, 2020).

We aim to gain an intuition on influence functions by studying their behavior in low dimensions on both synthetic and tabular data. We also develop heuristic algorithms which are simpler and less expensive to compute, which perform competitively with influence functions when used for finding influential groups of points.

3.1 Influence Function Theory

Influence functions, as introduced by Koh and Liang (2017), are given as follows. Suppose we have a prediction problem with input space $\mathcal{X}$ and output space $\mathcal{Y}$. Given a list of training points $Z_{\text{train}} = \{z_1, z_2, \ldots, z_n\}$ where $z_i = (x_i, y_i)$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$, we can train a parametric model with parameter class Θ with respect to a certain loss function L by finding the model parameters $\theta \in \Theta$ which minimize the empirical risk, which is given by $R(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$. Namely, the model training process is equivalent to finding

$$\hat{\theta} = \operatorname*{arg\,min}_{\theta \in \Theta} R(\theta) = \operatorname*{arg\,min}_{\theta \in \Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) \tag{3.1}$$

Note that we fold any regularization term into L. Then, given a test instance z_t, we define the influence of training instance z on z_t as the change in prediction probability for the predicted class of the model on input z_t when z is removed from Z_{train} and the model is retrained. While retraining is expensive in general, we can approximate this by considering an upweighting of the loss term corresponding to z by ϵ and considering the optimal parameters of this upweighted loss

$$\hat{\theta}_{\epsilon,z} = \operatorname*{arg\,min}_{\theta \in \Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta) \tag{3.2}$$

First, we will compute the change in these parameters with respect to ϵ, which is given by $\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0}$. Since $\hat{\theta}_{\epsilon,z}$ minimizes the upweighted empirical risk, we have

$$\nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}) = 0 \tag{3.3}$$

We define $\Delta_\epsilon \triangleq \hat{\theta}_{\epsilon,z} - \hat{\theta}$. Performing a first order Taylor expansion of equation 3.3 at $\hat{\theta}$ (noting that $\lim_{\epsilon\to 0} \hat{\theta}_{\epsilon,z} = \hat{\theta}$) gives

$$\left(\nabla R(\hat{\theta}) + \epsilon \nabla L(x,\hat{\theta})\right) + \left(\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(x,\hat{\theta})\right) \Delta_\epsilon \approx 0 \implies \tag{3.4}$$

$$\Delta_\epsilon \approx -\left(\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(x,\hat{\theta})\right)^{-1} \left(\nabla R(\hat{\theta}) + \epsilon \nabla L(x,\hat{\theta})\right) \approx -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z,\hat{\theta})\epsilon \tag{3.5}$$

where the last (approximate) equality comes from dropping $o(\epsilon)$ terms and using the fact that $\nabla R(\hat{\theta}) = 0$ (as $\hat{\theta}$ minimizes R). Using this, we can compute

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = \left.\frac{d\Delta_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z,\hat{\theta}) = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z,\hat{\theta}) \tag{3.6}$$

where $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i,\hat{\theta})$ is the empirical Hessian matrix. Assuming R is twice-differentiable and strongly convex in θ, $H_{\hat{\theta}}$ will be positive definite and thus invertible. Note that the first equality in equation 3.6 comes from the fact that $\hat{\theta}$ does not depend on epsilon. To get the final form of the influence function, which is the derivative of the loss on the test point z_t with respect to ϵ at $\epsilon = 0$, we apply the chain rule

$$\mathrm{Inf}(z, z_t) = \left.\frac{dL(z_t, \hat{\theta}_{\epsilon,z})}{d\epsilon}\right|_{\epsilon=0} = \nabla_\theta L(z_t,\hat{\theta})^\top \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -\nabla_\theta L(z_t,\hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z,\hat{\theta}) \tag{3.7}$$

3.2 Influence Function are Approximations

While influence functions aim to measure a model's prediction change on a test point when a training point is removed, they are ultimately an approximation. Influence functions drop several smaller order terms, make convergence assumptions, and also approximate group effects by directly combining individual effects.

3.2.1 Dropping Smaller Order Terms and Convergence Assumptions

There are several areas of approximation when using influence functions. First, influence functions use a first order Taylor expansion which drops both $o(\Delta_\epsilon)$ and $o(\epsilon)$ terms as given in equation (3.4). Additionally, computing the influence function uses the empirical risk minimizer $\hat{\theta}$. In practice, many models are trained using iterative algorithms, such as gradient descent based algorithms, and it is not guaranteed that the parameters returned from such a training procedure is the true $\hat{\theta}$. Finally, the derivation of the influence function in Section 3.1 assumes that the empirical risk is strongly convex in θ. While Koh and Liang (2017) show that computing the influence function using $\tilde{\theta}$ near $\hat{\theta}$ and on a non-convex objective still gives meaningful results in practice, their theoretical analysis introduces more approximations of dropping smaller order terms.

3.2.2 Approximating Group Effects

In addition to finding individual training points which are most influential on a test point for a given model, one may be interested in finding a group of training points of a certain size which, taken altogether, are the most influential on a test point (i.e. when removed and the model is retrained, the prediction confidence on the test point decreases the most). Searching over all such groups is intractable for most datasets and group sizes of practical interest, and Koh et al. (2019) investigate the use of influence functions on individual points to approximate a group effect. While the authors show empirically that influence function computations for groups of points is correlated with their actual effect, their theoretical analysis also shows that this correlation may not hold in general.

3.3 Related Work

Koh and Liang (2017) describe the assumptions and mathematical guarantees of influence functions, and a practical approximate way to implement them. To make the computation of influence tractable, they use the quadratic Taylor expansion approximation, which leads to the expression in equation 3.7. Additionally, if the empirical risk is not convex, or the parameters found in the model training process have not fully converged to the minimizer $\hat{\theta}$ in equation 3.1 (e.g. if a training procedure such as stochastic gradient descent is used with an early stopping), then the derivation of equation 3.7 may not hold. In this non-convex or not yet converged case, the authors demonstrate empirical results which show influence functions still give meaningful results in practice. However, we believe that such a case may require more scrutiny. Finally, the inverse empirical Hessian matrix $H_{\hat{\theta}}^{-1}$ is impractical to compute for large models with many parameters, and thus the authors result to stochastic estimation. Schioppa et al. (2021) use a trick to speed up the inverse Hessian calculation at the cost of using additional approximations.

Pruthi et al. (2020b) describe a method to compute influence of training instance z_i on test instance z by adding up the change in loss of z between iteration $t+1$ and iteration t for all time steps t for which the training instance z_i is used. However, to make this implementation practical, the authors result to an approximate computation which uses evenly spaced checkpoints, using the next iteration where z_i is used if it is not used at one of the checkpoints. It has been shown that this method, as well as the one used by Koh and Liang (2017), are unstable and very sensitive to factors such as initialization, training data order, and batch size (K and Søgaard, 2021), and are also fragile (Basu et al., 2020).

Most aligned with our work are works which use heuristics to speed up computation of approximate influence. Guo et al. (2021) use k-nearest neighbors as a heuristic to reduce the search space before applying more expensive influence function computations while Rajani et al. (2020) directly uses k-nearest neighbors in the embedding space of language models such as BERT and RoBERTa based systems to heuristically find the most influential training points on a given input instance for natural language tasks. Instead, we study logistic regression models trained on low dimensional data to develop geometric intuition and heuristics which can be used as cheaper alternatives to influence functions. We leave it to future work to extend this intuition to more complex models and higher dimensional data.

3.4 Developing Heuristics

We restrict our study to logistic regression models on binary classification tasks. We present several heuristics which can be used in place of influence functions for finding influential groups. While we do not prove any theoretical guarantees on these heuristics, we show empirically that their performance is competitive with influence functions both on synthetic and real world tabular data.

3.4.1 Developing Intuition

Greedy Algorithm for finding Influential Sets

As previously mentioned, finding the exact most influential set requires searching over all subsets of training points of the desired size, which is computationally infeasible for most datasets of practical interest. Thus, we use a beam search given in algorithm 1 to greedily build our approximately most influential set. We did not observe any differences for a beam size of up to 10 on our synthetic data, and increasing the beam size significantly increases the computation time of the search. Thus for computational reasons, we use a beam size of 1, which is equivalent to a simple greedy algorithm of, at each step, picking the training point which when combined with the current set, decrease the prediction confidence of the test point the most when these points are removed and the model is retrained. Note that the greedy algorithm is not a feasible approach to use over influence functions in practice due to computational reasons, as it requires retraining the model $O(nk)$ times, where n is the size of the training set and k is the desired size of the influential set.

Algorithm 1: Beam Search for Most Positively Influential Points

```
   Data: training set $Z$, beam size $b$, group size $k$, test point $z$
1  $B \leftarrow \emptyset$;
2  for $i = 1, \ldots, k$ do
3      $B' \leftarrow \emptyset$;
4      for $S \in B$ do
5          for $z' \in Z \setminus S$ do
6              $S' \leftarrow S \cup \{z'\}$;
7              if $|B'| < b$ then
8                  $B' \leftarrow B' \cup \{S'\}$
9              else
10                 $S^* \leftarrow \min_{T \in B'} score(T)$ ;    /* $score(T) = M_z(Z \setminus T) - M_z(Z)$
                    where $M(D)$ denotes the model's prediction for $z$
                    when trained on data $D$.  */
11                 if $score(S') > score(S^*)$ then
12                     $B' \leftarrow B \setminus \{S^*\}$;
13                     $B' \leftarrow B \cup \{S'\}$;
14     $B \leftarrow B'$;
15 return $B$
```

Maximal Rotation Intuition

We begin our investigation with a two dimensional Gaussian synthetic dataset $Z = \{z_1, \ldots, z_n\}$. For $1 \leq i \leq \frac{n}{2}$, we have $z_i = (x_i, y_i, l_i)$ where x_i, y_i are drawn independently from $N(2, \sigma^2)$ and the label is $l_i = 1$. For $\frac{n}{2} < i \leq n$, we have $z_i = (x_i, y_i, l_i)$ where x_i, y_i are drawn independently from $N(-2, \sigma^2)$ and the label is $l_i = 0$. We train a logistic regression model on this data. Applying algorithm 1 to this data, we see that a good heuristic for searching for a set of maximally influential points seems to be inducing a maximal rotation on the decision boundary toward the test point, with a further center of rotation being better (see figure 3-1 for an example of this). This intuition aligns with the fact that, assuming a fixed magnitude of coefficients, a logistic regression model's prediction probability on a (correctly predicted) data point is monotonically increasing in the distance of that data point to the model's decision boundary. In other words, if the data point is further from the decision boundary (and on the correct side), the logistic regression model will give a higher probability for the predicted label. To constrain the magnitude of the coefficients, we

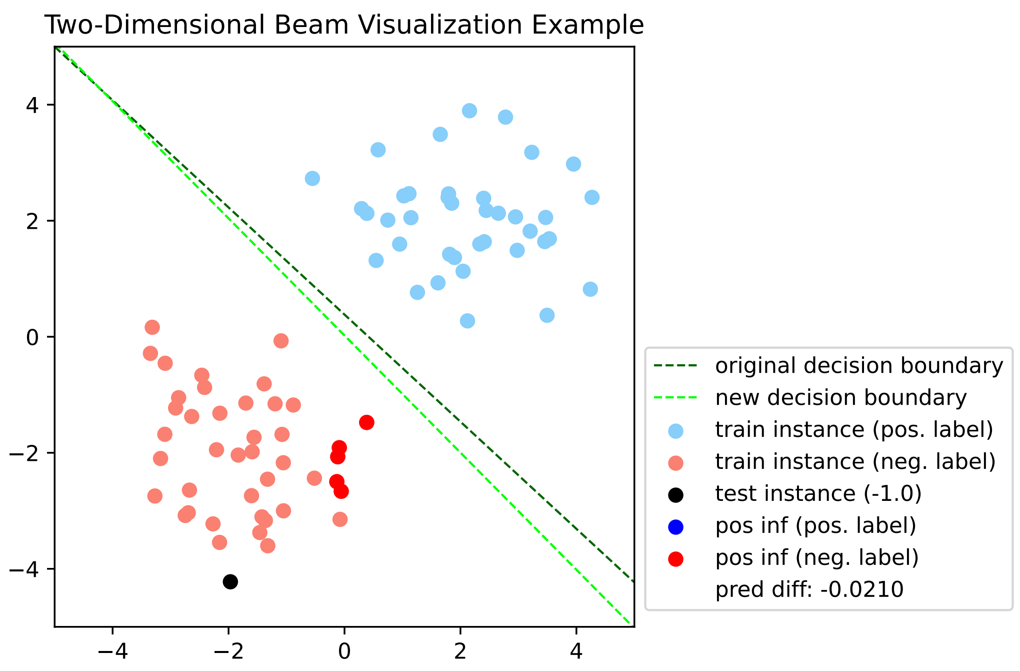

Figure 3-1: Scatter plot showing the most positively influential points selected by algorithm 1 with $b = 1$ and $k = 5$ for a logistic regression model trained on the data generated as described above with $\sigma^2 = 1$ and a negatively labeled test point generated from the same distribution as the training data.

add a regularization term to our model. Regularization is also helpful in general for logistic regression models, especially when the training data is linearly separable, as otherwise the weights will blow up when attempting to minimizing the cross entropy loss.

3.4.2 Heuristics

Nearest Neighbors of Projection onto Decision Boundary

With the intuition described in section 3.4.1 in mind, a simple heuristic for selecting an influential group of k points is taking the k nearest training points to the projection of the test point onto the decision boundary which share the same label as the test point. Call the test point x_t, its projection p_t, and the original model's decision boundary B, which is a vector. This heuristic will select points near the decision boundary, as well as points near x_t in the direction along the decision boundary. As points near the decision boundary will incur higher losses, their removal will help induce a rotation in the decision boundary better than points far away from the decision boundary. Additionally, intuitively in the two dimensional case, the removal of the most extreme points the direction along either B or $-B$, depending on where x_t is relative to the training data, will help induce a center of rotation furthest away from p_t. We will explore this intuition more in section 3.4.2. For now, we can see that using the nearest neighbors to p_t is a simple heuristic which will give a trade-off between these two competing pieces of intuition.

Failure of Influence Functions to Capture Group Effects

Consider the case where training data Z is symmetrically distributed around x_t. Then, if we use influence functions to find a positively influential set, we will be ambivalent in picking the points in the B direction and the $-B$ direction (as two points which are equally extreme in both these directions could have equal influence scores). However, picking points in both directions will not have as strong of a group effect as picking only points in one direction, as this will induce more of a rotation on the decision boundary. In this case, the marginal effects attributed to the training points by influence functions fail to completely capture an optimal group effect.

Perpendicular Post-processing in Two Dimensions

As mentioned in section 3.4.2, one heuristic is to consider the most extreme training points along either B or $-B$, depending on where x_t is relative to the training data. This is because removing the most extreme points in the B direction will move the center of rotation further in the $-B$ direction, and vice versa. Thus choosing to remove the most extreme points in the B or $-B$ direction comes down to whether x_t is closer to the extreme along B or $-B$ with respect to the training data. As a simple proxy metric for this, we pick which direction to use based on which side of B' has more training points, where B' is the perpendicular to B through x_t.

Putting these ideas into a heuristic post processing algorithm, we first run the nearest neighbors to projection heuristic described in section 3.4.2, which returns a set of training points P. Then, we compute a set of candidate training points C by taking all points in Z on the side of B' with less training points which also share the same label as x_t and are closer to the decision boundary than the furthest point from the decision boundary in P. We then iteratively try to replace the furthest points in P from the decision boundary with the most extreme points along B (or $-B$, depending on which side of B' has less points in Z). See algorithm 2 for details. See figure 3-2 (right) for an example of this heuristic, and how it improves upon figure 3-2 (left).

Algorithm 2: Post-processing for Most Positively Influential Points Based on Distance to Decision Boundary

```
   Data: training set $Z$, selected points $P$, decision boundary $B$, test point $z$
1  $P' \leftarrow$ copy of $P$;
2  $B' \leftarrow$ line through $z.x$ perpendicular to $B$;
3  sort $p \in P$ by $dist(p.x, B)$ in descending order;
4  $m \leftarrow \min_{p \in P} dist(p.x, B)$;
5  $C \leftarrow \{\}$;
6  $s \leftarrow$ side of $B'$ containing less points in $Z$;
7  for $p \in Z$ do
8      if $dist(p.x, B) \leq m$ and $side(p.x, B) = s$ and $p.y = z.y$ and $p \notin P$ then
9          $C \leftarrow C \cup \{p\}$;
10 sort $p \in C$ by $d(p, B')$ in descending order;
11 $i, j \leftarrow 0, 0$;
12 while $i < |P|$ and $j < |C|$ do
13     if $dist(C[j].x, B) < dist(P[i].x, B)$ then
14         $P' \leftarrow P' \setminus \{P[i]\}$;
15         $P' \leftarrow P' \cup \{C[j]\}$;
16         $i \leftarrow i + 1$;
17     $j \leftarrow j + 1$;
18 return $P'$
```

Parallel Post-processing in Two Dimensions

The post-processing procedure described in section 3.4.2 may be too conservative in some settings. For example, suppose that we want to pick extreme points in the B direction. If the most extreme points in the B direction are all further from the decision boundary than the furthest point in our current influential set (before

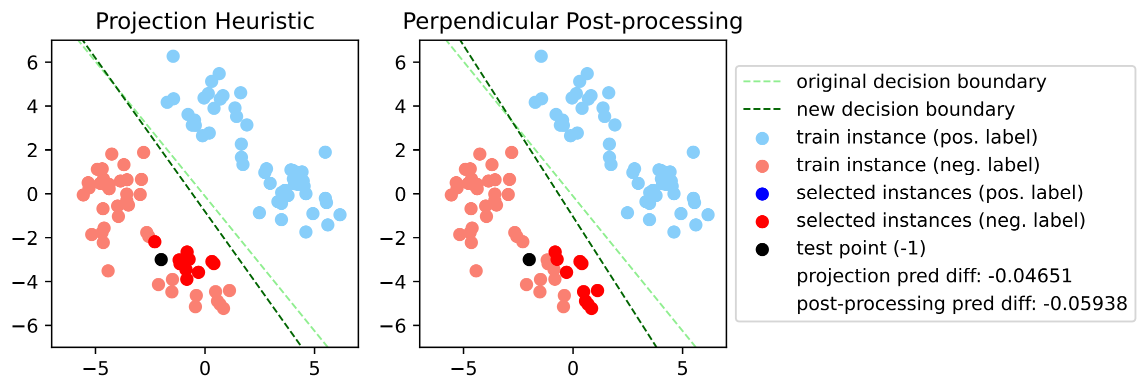

Figure 3-2: Scatter plot showing the most positively influential points selected by running algorithm 2 with selected points input P as the set returned from running the projection heuristic described in section 3.4.2. The data is generated according to a Gaussian mixture distribution described in section 3.5.1 with a training size of 100, influential set size of 10, and $\sigma^2 = 1$.

post-processing) is from the decision boundary, then we will not select any of these points. Another post-processing perspective is to replace points on the undesired side of x_t with the most extreme training points on the desired side of x_t whenever possible. Details of this post-processing algorithm are given in algorithm 3. Because this algorithm, as given, does not prevent training points very far from the decision boundary to be chosen, we also consider a variant which only considers points which are closer to the decision boundary than the test point is in the post-processing step. See figure 3-3 for an example.

Same Side Choice

When dealing with data which is not linearly separable, it may be beneficial to constrain the points selected by the heuristics we presented above. Thus, we also consider variants of each of the heuristics presented in the above sections where we only consider training points on the same side of the decision boundary as x_t for potential inclusion in a positively influential set. For some intuition behind this, consider the following scenario. Test point x_t is negatively labeled and the decision boundary correctly places it on the negative side. There is a set of several negatively labeled points N near the decision boundary but far from each other, and correctly classified

Algorithm 3: Post-processing for Most Positively Influential Points Based on Distance Along Decision Boundary

```
Data: training set $Z$, selected points $P$, decision boundary $B$, test point $z$
1  $B' \leftarrow$ line through $z.x$ perpendicular to $B$;
2  $m \leftarrow \min_{p \in P} dist(p.x, B)$;
3  $s \leftarrow$ side of $B'$ containing less points in $Z$;
4  $Q \leftarrow \{\}$;
5  for $p \in P$ do
6      if $side(p.x, B) \neq s$ then
7          $Q \leftarrow Q \cup \{p\}$;
8  sort $p \in Q$ by $d(p, B')$ in descending order;
9  $C \leftarrow \{\}$;
10 for $p \in Z$ do
11     if $side(p.x, B) = s$ and $p.y = z.y$ and $p \notin P$ then
12         $C \leftarrow C \cup \{p\}$;
13 sort $p \in C$ by $d(p, B')$ in descending order;
14 $i, j \leftarrow 0, 0$;
15 $P' \leftarrow$ copy of $P$;
16 while $i < |Q|$ and $j < |C|$ do
       /* Variant: Only update $P'$ if $dist(C[j], B) < dist(z.x, B)$ */
17     $P' \leftarrow P' \setminus \{Q[i]\}$;
18     $P' \leftarrow P' \cup \{C[j]\}$;
19     $i \leftarrow i + 1$;
20     $j \leftarrow j + 1$;
21 return $P'$
```

(i.e. on the correct side of the decision boundary). Now, say there is one more negatively labeled point p which is on the positive side of the decision boundary, but near N. Even though p may be selected by our heuristic algorithms (or picked up during post-processing), its removal may not help much in inducing a rotation of the decision boundary toward x_t, if at all, as a rotation would move the decision boundary closer to N, which would increase the loss incurred by the points in N.

Extension to Higher Dimensions

The nearest neighbors of projection onto decision boundary heuristic, as described in section 3.4.2, extends easily into higher dimensions. However, the rotation intuition becomes more complicated. We leave exploring geometric intuition for this heuristic in higher dimensions to future work. While the post-processing heuristics only make

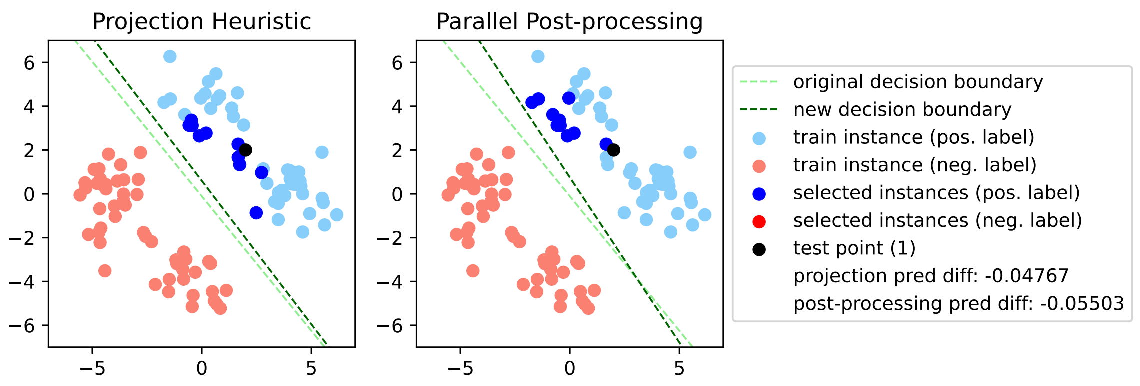

Figure 3-3: Scatter plot showing the most positively influential points selected by running algorithm 3 (the variant only considering candidates closer to the decision boundary than the test point) with selected points input P as the set returned from running the projection heuristic described in section 3.4.2. The data is generated according to a Gaussian mixture distribution described in section 3.5.1 with a training size of 100, influential set size of 10, and $\sigma^2 = 1$.

sense in two dimensions, we can still apply these heuristics in the case of binary classification on higher dimensional data by first mapping the data to two dimensions using a linear dimensionality reduction method such as principal component analysis (PCA). We can then train a logistic regression model on the reduced two dimensional data and apply the post-processing heuristics. Because PCA is a linear dimensionality reduction method, the maximal rotation intuition carries over, but projected onto the two principal components of the original data.

Run Time Comparison

Computing the influence function for a given test point takes $O(np^2 + p^3)$ where n is the number of training points and p is the number of features. In comparison, the projection heuristic discussed in section 3.4.2 takes $O(np)$ time to compute. When $p = 2$, both of these are $O(n)$. Additionally, each of the post-processing algorithms also run in $O(n)$ when $p = 2$ as we are iterating through each training point at most once, and in each iteration we do a constant number of comparisons. In general, as mentioned above, using the post-processing algorithms requires an initial PCA to map the data to two dimensions, which takes $O(\max(n, p)^2 \cdot \min(n, p))$ using a full singular value decomposition (SVD) solver, but this is is reduced to $O(\max(n,p)^2)$ using a randomized solver (Pedregosa et al.). See table 3.1 for a comparison of the run times of the different approaches.

Using the Best of Multiple Heuristics

Each of the heuristics mentioned above may work better in certain cases, depending on the data. Thus, we also consider running each of the heuristics, and trying each of the influential sets we get to see which one's removal maximally decreases the prediction confidence of a test point of interest. Because each post-processing heuristic has the same run time, and we try a constant number of them, using the best of multiple heuristics will not increase the asymptotic run time. Note that the time to retrain the model must also be factored into this, but since training a logistic regression model takes $O(np)$ time, it does not increase the asymptotic run time of using the best of multiple heuristics.

Influence Function	$O(np^2 + p^3)$
Projection Heuristic	$O(np)$
Best of Heuristics in 2D using PCA	$O(\max(n,p)^2)$
Best of Heuristics and Influence Function in 2D using PCA	$O(\max(n,p)^2)$

Table 3.1: Asymptotic run times of different approaches in the general case. The run times of the two approaches using PCA use the run time of the randomized SVD solver.

3.5 Heuristic Experiments and Results

We compare the performance of influence functions, the greedy algorithm, and the heuristics described in section 3.4 on synthetic and tabular datasets.

3.5.1 Datasets

Gaussian Data

As described in section 3.4.1, the first dataset we look at is Gaussian data where the positively labeled points are generated via $x_1, x_2 \overset{\text{iid}}{\sim} N(2, \sigma^2)$ and the negatively labeled points are generated via $x_1, x_2 \overset{\text{iid}}{\sim} N(-2, \sigma^2)$. We set $\sigma^2 = 1$ to get a linearly separable dataset and $\sigma^2 = 2$ to get a dataset which is not linearly separable. We generate a balanced training set of size 400 and test point from the same distribution (with probability $\frac{1}{2}$ of being from each class), and compute the prediction drop of the test point when the influential sets of size 40 (10% of the training set size), as found by the greedy approach, influence function approach, and heuristic approaches, are removed and the model is retrained. We average this prediction drop over 100 trials (where we regenerate both the training data and test data in each trial).

Gaussian Mixture Data

Next, we consider Gaussian mixture data. We generate positively labeled points as follows. First generate $p \sim \text{Bernoulli}(\frac{1}{2})$. Then we generate $x_1, \sim N(4, \sigma^2), x_2, \sim N(0, \sigma^2)$ if $p = 0$ and $x_1, \sim N(0, \sigma^2), x_2, \sim N(4, \sigma^2)$ if $p = 1$. To generate negatively labeled points, we again generate $p \sim \text{Bernoulli}(\frac{1}{2})$. Then we generate $x_1, \sim N(-4, \sigma^2), x_2, \sim N(0, \sigma^2)$ if $p = 0$ and $x_1, \sim N(0, \sigma^2), x_2, \sim N(-4, \sigma^2)$ if $p = 1$. Again, we set $\sigma^2 = 1$ to get a linearly separable dataset and $\sigma^2 = 2$ to get a dataset which is not linearly separable. We again generate a balanced training set of size 400 and test point from the same distribution (with probability $\frac{1}{2}$ of being from each class), and compute the prediction drop of the test point when the influential sets of size 40 (10% of the training set size), as found by the greedy approach, influence function approach, and heuristic approaches, are removed and the model is retrained. We also average this prediction drop over 100 trials (where we regenerate both the training data and test data in each trial).

Banknote Authentication Dataset

The banknote authentication dataset is a tabular dataset consisting of binary labeled data (representing if the note is genuine or forged) with four real, continuous features (Dua and Graff, 2017). We take a training set size of 400 randomly selected samples and a test set size of 100 randomly selected samples. We compute the prediction drop of each test point when the influential sets of size 40 (10% of the training set size), as found by the greedy approach, influence function approach, and heuristic approaches, are removed and the model is retrained. We average this prediction drop over all 100 test samples and compare results. We also compare approaches on a two dimensional version of the banknote authentication dataset, where we first project the entire dataset into two dimensions using PCA.

Wine Quality Dataset

The wine quality dataset is a tabular dataset consisting of 11 classes (quality score from 0 to 10) with 11 real, continuous features (Cortez et al., 2009; Dua and Graff, 2017). The dataset contains data for both red and white wine. For both wine types, we turn the tasks into binary classification by removing all data with quality 6 and predicting quality ≤ 5 against quality ≥ 7. We then balance the dataset by randomly selecting 1000 samples per class. Next, we take a training set size of 400 randomly selected samples and a test set size of 100 randomly selected samples. We compute the prediction drop of each test point when the influential sets of size 40 (10% of the training set size), as found by the greedy approach, influence function approach, and heuristic approaches, are removed and the model is retrained. We average this prediction drop over the 100 test samples and compare results.

3.5.2 Results

The results of the best performing heuristics, as well as from influence function and greedy, are given in table 3.2. We also compare to a baseline heuristic, which is simply taking the nearest points to the decision boundary with the same label as the test point. Note that for the post-processing heuristics (perp. and par. columns) on the banknote data, we first project to two dimensions using PCA and then run these heuristics to find the influential set. We obtain the prediction drop by retraining a model on the full data with this influential set removed (where we map back to the full data). In case of the two dimensional data, as well as in the case of the banknote data, the difference in prediction drops between the influence function and the best of heuristics is never more than 0.6%. In the case of the wine data, this difference is bigger, but still does not exceed 4%. This suggests that the heuristics are competitive with the influence function, especially in the lower dimensional settings.

	Near to Bound.	Proj.	Perp.	Par. (cutoff at test pt)	Best	Inf. Func.	Greedy
Gaussian Separable	-4.29%	-4.39%	-4.44%	-4.18%	-4.44%	-4.45%	-4.48%
Gaussian Non-separable	-6.29%	-7.85%	-8.35%	-8.21%	-8.45%	-8.49%	-9.01%
Mixture Separable	-4.34%	-6.35%	-6.57%	-6.25%	-6.58%	-6.59%	-6.75%
Mixture Non-separable	-6.04%	-9.15%	-10.3%	-9.68%	-10.3%	-10.5%	-11.2%
Banknote 2D (via PCA)	-5.59%	-10.5%	-11.3%	-11.1%	-12.8%	-13.4%	-14.1%
Banknote	-3.30%	-9.03%	-7.31%	-8.17%	-11.8%	-12.3%	-13.2%
Red Wine	-5.95%	-9.03%	-9.64%	-9.83%	-16.8%	-18.7%	-21.7%
White Wine	-6.05%	-11.0%	-10.6%	-12.7%	-17.0%	-20.9%	-22.2%

Table 3.2: Comparison of different heuristics with influence function and greedy. For postprocessing, we take the best performance of whether or not to use only training points on the same side of the decision boundary as the test point. For the banknote, red wine, and white wine data, the postprocessing is done by projecting the data into two dimensions using PCA before running the heuristics on the two dimensional data. In these cases, the best column considers the two dimensional influence function as well (as the influence functions are cheap to compute on the two dimensional data compared to the full data).

Chapter 4

Conclusion

In this thesis, we explore two types of model interpretability techniques. First, we take a closer look at neural rationale models, which use of a selector and classifier framework, with the selector extracting a segment of the input text to pass to the classifier. As the extracted segment serves as a bottleneck containing the only information the classifier uses, they have been claimed to be "inherently interpretable". However, we argue against the commonly held belief that rationale models are inherently interpretable by design. We present several reasons, including a counter-example showing that a reasonable-looking model could be as non-interpretable as a black-box. These reasons imply that the missing piece is an understanding of the *rationale selection process* (i.e. the selector). We also conduct a (non-adversarial) perturbation-based study to investigate the selector of two rationale models, in which automated analyses and a user study confirm that they are indeed hard to understand. In particular, the higher-accuracy model (CR) fares worse in most aspects, possibly hinting at the performance-interpretability trade-off (Gunning and Aha, 2019). These results point to a need for more rigorous analysis of interpretability in neural rationale models.

Next, we turn our focus to influence functions, which are a type of training based explanation method. Unlike rationale models which have a specific architecture containing a selector and classifier with the selector's output serving as an explanation, training based explanation methods provide explanation for an arbitrary model's output on a given test point by finding a set of most influential training points for that test point. In the case of influence functions, influence of a training point is measured by (approximately) computing the model's prediction difference on the test point when the training point is removed and the model is retrained. We present several cheap to compute heuristics based on geometric intuition in the case of a logistic regression model on two dimensional binary classification to find sets of influential training points. While the geometric intuition is based on two dimensional settings, we extend the heuristics to higher dimensional tabular data via PCA. Finally, we show via experiments on both synthetic data and tabular data, our heuristics perform competitively with influence functions. Future directions involve searching for a set of mathematical rules based on the shape of the training data to determine which heuristic to use, which would prevent the need from trying all heuristics to find the best performing one for a given training set and test point. Future directions also include extending geometric intuition to higher dimensions more directly (e.g. without having to use a dimension reduction method), and extending heuristics to nonlinear classifiers such as neural networks.

Appendix A

Neural Rationale Experiment Details

A.1 Additional Details on the Experimental Setup

A.1.1 Training

The models we train are as implemented in (Bastings et al., 2019). The hyperparameters we use are 30 percent for the word selection frequency when training the CR model and L_0 penalty weight 0.01505 when training the PG model. Training was done on a MacBook Pro with a 1.4 GHz Quad-Core Intel Core i5 processor and 8 GB 2133 MHz LPDDR3 memory. The training time for each model was around 15 minutes. There are a total of 7305007 parameters in the CR model and 7304706 parameters in the PG model. The hyperparameter for the CR model is the word selection frequency, ranging from 0% to 100%, whereas the hyperparameter for the PG model is the L_0 penalty weight which is a nonnegative real number (for penalizing gaps in selections).

These hyperparameter were configured with the goal that both models would select a similar fraction of total words as rationale. This was done manually. Only one CR model was trained (with the word selection frequency set to 30 percent). Then, a total of 7 PG models were trained, with L_0 penalty weight ranging from 0.01 to 0.025. Then, the closest matching result to the CR model in terms of word selection fraction, which was an L_0 penalty of 0.01505, was used.

The CR model (with 30% word selection frequency) achieves a 47.3% test accuracy with a 24.9% rationale selection rate, and the PG model (with L_0 penalty of 0.01505) achieves a 43.3% test accuracy with a 23.1% selection rate, consistent with those obtained by Bastings et al. (2019, Figure 4). The CR model achieves a validation accuracy of 46.0% with a 25.1% rationale selection rate, and the PG model achieves a 41.1% validation accuracy with a 22.9% selection rate, comparable to the test results.

A.1.2 Dataset

We use the Stanford Sentiment Treebank (SST, Socher et al., 2013) dataset with the exact same preprocessing and train/validation/test split as given by Bastings et al. (2019). There are 11855 total entries (each are single sentence movie reviews in English), split into a training size of 8544, a validation size of 1101, and a test size of 2210. The label distribution is 1510 sentences of label 0 (strongly negative), 3140 of label 1 (negative), 2242 of label 2 (neutral), 3111 of label 3 (positive), and 1852 of label 4 (strongly positive). We use this dataset as is, and no further pre-processing is done. The dataset can be downloaded from the code provided by Bastings et al. (2019).

A.1.3 Sentence Perturbation

The data perturbation was done on the same machine with specs described in Appendix A.1. This procedure was done once and took around an hour. This perturbation was an automated procedure using the BERT and WordNet synset intersection as a heuristic for word substitutions. As a result, we did not collect any new data which requires human annotation or other work.

A.2 Additional Examples of Sentence Perturbation

Table A.1 shows ten randomly sampled perturbations.

There are weird resonances between actor and **role**/*character* here , and they 're not exactly flattering .
A loving **little**/*short* film of considerable appeal .
The film is really not so **much**/*often* bad as bland .
A cockamamie tone poem pitched precipitously between swoony lyricism and violent catastrophe ... the most aggressively nerve-wracking and screamingly neurotic romantic comedy in **cinema**/*film* history .
Steve Irwin 's method is Ernest Hemmingway at accelerated speed and **volume**/*mass* .
The movie addresses a hungry need for PG-rated , nonthreatening family **movies**/*film* , but it does n't go too much further .
... the last time I saw a theater full of people constantly checking their **watches**/*watch* was during my SATs .
Obvious politics and rudimentary animation reduce the **chances**/*chance* that the appeal of Hey Arnold !
Andy Garcia enjoys one of his richest roles in years and Mick Jagger gives his best **movie**/*film* performance since , well , Performance .
Beyond a handful of mildly amusing lines ... there just **is**/*be* n't much to laugh at .

Table A.1: Ten randomly sampled sentence perturbation examples given in a user study, with the original word shown in **bold** replaced by the word in *italics*.

A.3 Description of the Human Evaluation of Data Perturbation

We recruited five graduate students with ML experience (but no particular experience with interpretable ML or NLP), and each participant was asked to answer questions for 20 sentence perturbations, for a total of 100 perturbations. An example question is shown below:

The original sentence (a) and the perturbed sentence (b), as well as the selected rationale on the original sentence (in bold) are:

- a There **are weird resonances** between actor and role here , and they **'re** not **exactly flattering** .
- b There are weird resonances between actor and character here , and they 're not exactly flattering .

The original prediction is: negative.

1. Should the prediction change, and if so, in which way:
2. If yes:
   - (a) Does the changed word need to be included or removed from the rationale?
   - (b) Please highlight the new rationale in red directly on the new sentence.

The study takes less than 15 minutes, is conducted during normal working hours with participants being grad students on regular stipends, and is uncompensated.

A.4 Additional Rationale Change Examples

Table A.2 shows additional rationale change examples.

PG	This delicately observed **story**/*tale* , deeply felt and masterfully stylized , is a triumph for its maverick director.
PG	Biggie and Tupac is so single-mindedly daring , it **puts**/*put* far more polished documentaries to shame.
PG	Somewhere short of Tremors on the modern B-scene : neither as funny nor as clever , though an agreeably unpretentious way to **spend**/*pass* ninety minutes .
PG	The film overcomes the regular minefield of coming-of-age cliches with **potent**/*strong* doses of honesty and sensitivity .
PG	As expected , Sayles ' smart wordplay and clever plot contrivances are as sharp as ever , though they may be overshadowed by some **strong**/*solid* performances .
CR	The animated subplot keenly depicts the inner **struggles**/*conflict* of our adolescent heroes - insecure , uncontrolled , and intense .
CR	Funny and , at times , poignant , the film from director George Hickenlooper all **takes**/*take* place in Pasadena , " a city where people still read . "
CR	It would be hard to think of a recent movie that **has**/*have* worked this hard to achieve this little fun.
CR	This road movie **gives**/*give* you emotional whiplash , and you 'll be glad you went along for the ride .
CR	If nothing else , this movie introduces a promising , unusual **kind**/*form* of psychological horror .

Table A.2: Additional rationale change example. Words selected in the original only, perturbed only, and both are shown in red, blue, and green, respectively.

A.5 Description of the User Study on Rationale Change

Participants were first given 10 examples of rationale selections (shown in bold) on the original and perturbed sentence pair made by the model, with one shown below:

orig: **Escapism** in its **purest** form .

pert: **Escapism** in its **purest** **kind** .

Then, they were presented with 20 test questions, where each question had two rationale assignments, one correct and one mismatched, and they were asked to determine which was the correct rationale assignment. An example is shown below:

a orig: **Benefits** from a **strong performance** from Zhao , but it 's Dong Jie 's **face** you **remember** at the end .

pert: Benefits from a **solid** performance from Zhao , but it 's Dong Jie 's **face** you **remember** at the end

b orig: Benefits from a **strong** performance from Zhao , but it 's Dong Jie 's **face** you **remember** at the end .

pert: **Benefits** from a **solid performance** from Zhao , but it 's Dong Jie 's **face** you **remember** at the end

In your opinion, which pair (a or b) shows the actual rationale selection by the model?

In the end, we ask the participants the following question for any additional feedback.

Please briefly describe how you made the decisions (which could include guessing), and your impression of the model's behavior.

The study takes less than 15 minutes, is conducted during normal working hours with participants being grad students on regular stipends, and is uncompensated.

Bibliography

Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, 2021.

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in neural information processing systems*, volume 31, pages 9505–9515, 2018.

Chirag Agarwal and Anh Nguyen. Explaining an image classifier's decisions using generative models. *arXiv preprint arXiv:1910.04256*, 10, 2019.

Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, page 559–560, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357944.

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

Anna Markella Antoniadi, Yuhan Du, Yasmine Guendouz, Lan Wei, Claudia Mazo, Brett A Becker, and Catherine Mooney. Current challenges and future opportunities for xai in machine learning-based clinical decision support systems: a systematic review. *Applied Sciences*, 11(11):5088, 2021.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile, 2020.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 648–657, 2020.

Serena Booth, Yilun Zhou, Ankit Shah, and Julie Shah. Bayes-trex: a bayesian sampling approach to model transparency by example. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

Kirill Bykov, Anna Hedström, Shinichi Nakajima, and Marina M-C Höhne. Noisegrad—enhancing explanations by introducing stochasticity to model weights. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6132–6140, 2022.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. *arXiv preprint arXiv:1807.08024*, 2018.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

Gilad Cohen, Guillermo Sapiro, and Raja Giryes. Detecting adversarial samples using influence functions and nearest neighbors, 2019.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009. ISSN 0167-9236. Smart Business Networks: Concepts and Empirical Evidence.

Ian C Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *The Journal of Machine Learning Research*, 22 (1):9477–9566, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and*

Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. Eraser: A benchmark to evaluate rationalized nlp models, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Ingo Feinerer and Kurt Hornik. *wordnet: WordNet Interface*, 2020. R package version 0.1-15.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*, 2018.

C Frye, D de Mijolla, L Cowton, M Stanley, and I Feige. Shapley-based explainability on the data manifold, arxiv. *arXiv preprint arXiv:2006.01272*, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning, 2019.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019.

Gary SW Goh, Sebastian Lapuschkin, Leander Weber, Wojciech Samek, and Alexander Binder. Understanding integrated gradients with smoothtaylor for deep neural network attribution. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4949–4956. IEEE, 2021.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

David Gunning and David Aha. Darpa's explainable artificial intelligence (xai) program. *AI Magazine*, 40(2):44–58, 2019.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Giles Hooker, Lucas Mentch, and Siyu Zhou. Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance. *Statistics and Computing*, 31:1–16, 2021.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *CoRR*, abs/1806.10758, 2018.

Alon Jacovi and Yoav Goldberg. Aligning faithful interpretations with their social attribution. *CoRR*, abs/2006.01067, 2020.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. *CoRR*, abs/1902.10186, 2019.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, 2020.

Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on artificial intelligence and statistics*, pages 2907–2916. PMLR, 2020.

Karthikeyan K and Anders Søgaard. Revisiting methods for finding influential examples, 2021.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 267–280, 2019.

Vid Kocijan and Samuel R Bowman. Influence functions do not seem to predict usefulness in nlp transfer learning, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2017.

Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects, 2019.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association for Computational Linguistics.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

Edward Loper and Steven Bird. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028, 2002.

Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

Andreas Madsen, Siva Reddy, and Sarath Chandar. Post-hoc interpretability for neural nlp: A survey. *arXiv preprint arXiv:2108.04840*, 2021.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

Luke Merrick and Ankur Taly. The explanation game: Explaining machine learning models using shapley values. In *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*, pages 17–38. Springer, 2020.

Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Decomposing signals in components (matrix factorization problems). `https://scikit-learn.org/stable/modules/decomposition.html#pca`.

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C. Lipton, Graham Neubig, and William W. Cohen. Evaluating explanations: How much do explanations from the teacher aid students? *CoRR*, abs/2012.00893, 2020a.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracing gradient descent, 2020b.

Nazneen Fatema Rajani, Ben Krause, Wengpeng Yin, Tong Niu, Richard Socher, and Caiming Xiong. Explaining and improving model behavior with k nearest neighbor representations, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*, 2020.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16:1–85, 2022.

Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 5–22, 2019.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference* *on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

Mike Wallace. *Jawbone Java WordNet API*, 2007.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi S Jaakkola. Rethinking cooperative rationalization: Introspective extraction and complement control. *arXiv preprint arXiv:1910.13294*, 2019.

Wei Zhang, Ziming Huang, Yada Zhu, Guangnan Ye, Xiaodong Cui, and Fan Zhang. On sample based explanation methods for nlp:efficiency, faithfulness, and semantic evaluation, 2021a.

Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5):726–742, 2021b.

Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. Do feature attribution methods correctly attribute features? In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. AAAI, Feb 2022a.

Yilun Zhou, Marco Tulio Ribeiro, and Julie Shah. Exsum: From local explanations to model understanding, 2022b.

Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. HILDIF: Interactive debugging of NLI models using influence functions. In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 1–6, Online, August 2021. Association for Computational Linguistics.