Combining Feature and Instance Attribution to Detect Artifacts

Anonymous ACL submission

Abstract

Training the deep neural networks that dominate NLP requires large datasets. These are often collected automatically or via crowdsourcing, and may exhibit systematic biases or *annotation artifacts*. By the latter we mean spurious correlations between inputs and outputs that do not represent a generally held causal relationship between features and classes; models that exploit such correlations may appear to perform a given task well, but fail on out of sample data. In this paper we evaluate use of different *attribution* methods for aiding identification of training data artifacts. We propose new hybrid approaches that combine *saliency maps* (which highlight "important" input features) with *instance attribution* methods (which retrieve training samples "influential" to a given prediction). We show that this proposed *training-feature attribution* can be used to efficiently uncover artifacts in training data when a challenging validation set is available. We also carry out a small user study to evaluate whether these methods are useful to NLP researchers in practice, with promising results.

1 Introduction

Large pre-trained (masked) language models dominate NLP leaderboards and are increasingly deployed in applications. But what exactly are such models "learning"? One concern is that they may be exploiting *artifacts* or spurious correlations between inputs and outputs that are present in the training data, but not reflective of the underlying task that the data is intended to represent.

We assess the utility of *attribution methods* for purposes of aiding practitioners in identifying training data artifacts, drawing inspiration from prior efforts that have suggested use of such methods for this purpose (Han et al., 2020; Zhou et al., 2021).

Warning: This paper contains examples with texts that might be considered offensive.

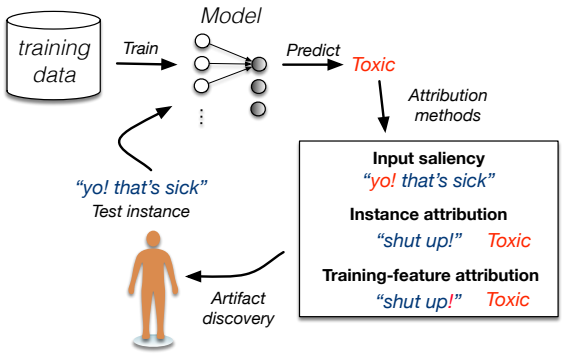

Figure 1: Use of different attribution techniques for artifact discovery in train data. Here attribution methods can reveal inappropriate reliance on certain tokens (e.g., *"!", "yo"*) to predict Tweet toxicity; these are artifacts.

Attribution methods are *model-centric*; our evaluation of them for artifact discovery therefore complements recent work on *data-centric* approaches to this end (Gardner et al., 2021). We consider two families of attribution methods: (1) *feature-attribution*, in which one highlights constituent input features (e.g., tokens) in proportion to their "importance" for an output (Ribeiro et al., 2016; Lundberg and Lee, 2017; Adebayo et al., 2018), and; (2) *instance attribution*, where one retrieves training instances most responsible for a given prediction (Koh and Liang, 2017; Yeh et al., 2018; Rajani et al., 2020; Pezeshkpour et al., 2021).

We also introduce new hybrid attribution methods that surface relevant *features within train instances* as an additional means to probe what the model has distilled from training data. This addresses inherent limitations of using either feature or instance attribution alone for artifact discovery: The former can only highlight patterns that in a given input, and the latter requires one to inspect entire (potentially lengthy) training instances to divine what might have rendered them influential.

Consider Figure 1. Here a model has learned to erroneously associate African American Vernacular English (AAVE) with *toxicity* (Sap et al., 2019)

and with certain punctuation marks ("!"). For a hypothetical test instance "yo! that's sick", both input saliency and instance attribution methods may provide some indication of these artifacts. But combining these via *training-feature attribution* (TFA) can directly surface the punctuation artifact by highlighting "!" within a relevant training example ("shut up!"); this is not readily apparent from either input or instance attribution. Our goal in this work is to evaluate TFA and other attribution methods as tools for identifying dataset artifacts.

Contributions. The main contributions of this paper are as follows. (1) We propose a new hybrid attribution approach, training-feature attribution (TFA), which addresses some limitations of existing attribution methods. (2) We evaluate feature, instance and training-feature attribution for artifact detection on several NLP benchmarks with previously reported artifacts to evaluate whether and to what degree methods successfully recover these, finding evidence that TFA can outperform other methods. We also discover and report previously unknown artifacts on a few datasets. Finally, (3) we conduct a small user-study to evaluate TFA for aiding artifact discovery in practice, and again find that combining feature and instance attribution is more effective at detecting artifacts than using either method on its own.

2 Background and Notation

Assume a text classification setting where the aim is to fit a classifier ϕ that maps inputs $x_i \in \mathcal{X}$ to labels $y_i \in \mathcal{Y}$. Denote the training set by $\mathcal{D} = \{z_i\}$ where $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Each x_i consists of a sequence of tokens $\{x_{i,1}, \ldots, x_{i,n_i}\}$. Here we define a linear classification layer on top of BERT (Devlin et al., 2019) as ϕ, fine-tuning this on $\mathcal{D}$ to minimize cross-entropy loss $\mathcal{L}$. Two types of attribution methods have been used in prior work to characterize the predictive behavior of ϕ.

Feature attribution methods highlight *important features* (tokens) in a test sample x_t. Examples of feature attribution methods include input gradients (Sundararajan et al., 2017; Ancona et al., 2018), and model-agnostic approaches such as LIME (Ribeiro et al., 2016). In this work, we consider only gradient-based feature attribution.

Instance attribution methods retrieve training samples z_i deemed "influential" to the prediction made for a test sample x_t: $\hat{y}_t = \phi(x_t)$. Attribution methods assign scores to train instances z_i intended to reflect a measure of importance with respect to $\hat{y}_t$: $I(\hat{y}_t, z_i)$. Importance can reflect a formal approximation of the change in $\hat{y}_t$ when z_i is upweighted (Koh and Liang, 2017) or can be derived via heuristic methods (Pezeshkpour et al., 2021; Rajani et al., 2020). While prior work has considered these attribution methods for "train set debugging" (Koh and Liang, 2017; Han et al., 2020), this relies on the practitioner to abstract away potential patterns within the influential instances.

3 Artifact Detection and Training-Feature Attribution

3.1 What is an *Artifact*?

Models will distill observed correlations between training inputs and their labels. In practice, some of these correlations will be *spurious*, by which we mean specific to the training dataset used. Consider a particular feature function f (such that $f(x)$ is 1 if x exhibits the feature extracted by f and 0 otherwise), a *training* distribution $\mathcal{D}$ over labeled instances z (often assembled using heuristics and/or crowdsourcing), and an ideal, hypothetical *target* distribution $\mathcal{D}*$ (the task we would actually like to learn; "sampling" directly from this is typically prohibitively expensive). Then we say that f is a *dataset* artifact if there exists a correlation between y and $f(x)$ in $\mathcal{D}$, but not in $\mathcal{D}*$. That is, if the mechanism by which one samples train instances induces a correlation between f and labels that would not be observed in an idealized case where one samples from the "true" task distribution.[1]

A given model may or may not exploit a particular dataset artifact; in some cases a *model-centered* view of artifacts may therefore be helpful. To accommodate this, we can extend our preceding definition by considering the relationship between model predictions $\hat{p}(y|x)$ and true conditional distributions $p(y|x)$ under D^*; we are interested in cases where the former differs from the latter due to exploitation of a dataset artifact f. Going further, we can ask whether this artifact was exploited *for a specific prediction*.

In this work we consider two types of artifacts. *Granular* input features refer to discrete units, such as individual tokens (this is similar to the definition of artifacts introduced in recent work by Gardner

[1] As a proxy for realizing this, imagine enlisting well-trained annotators with all relevant domain expertise to label instances carefully sampled i.i.d. from the distribution from which our test samples will actually be drawn in practice.

et al. 2021). *Abstract* features refer to higher-level *patterns* observed in inputs, e.g., lexical overlap between the premise and hypothesis in the context of NLI (McCoy et al., 2019).

3.2 Training-Feature Attribution

Showing important training instances to users for their interpretation places the onus on them to determine *what* was relevant about these instances, i.e., which features (granular or abstract) in x_i were influential. To aid artifact detection, it may be preferable to automatically highlight the tokens most responsible for the influence that train samples exert, communicating *what made an important example important*. This hybrid training-feature attribution (TFA) can reveal patterns extracted from training data that influenced a test prediction, even where the test instance does not itself exhibit this pattern, whereas feature attribution can only highlight features within said test instance. And unlike instance attribution, which retrieves entire train examples to be manually inspected (a potentially time-consuming and difficult task), TFA may be able to succinctly summarize patterns of influence.

A high-level schematic of TFA is provided in Figure 2. We aim to trace influence back to features within training samples. Koh and Liang (2017) used gradients to identify influential features within a training point z_i: $\nabla_{x_i} \mathrm{I}(x_t, z_i)$. We introduce TFA by extending this to extract influential features from training samples for a specific test prediction by considering alternative combinations of feature and instance attribution and means of aggregating over these as TFA variants. After calculating the importance of features within a train sample for a test target, we either apply the heatmap approach to identify *abstract* artifacts, or incorporate aggregate strategies to detect *granular* artifacts (which we describe below) and present them to users.[2]

Heatmaps We present the top and bottom k influential examples to users with *token highlights* communicating the relative importance of tokens within these k influential train instances. This may allow practitioners to interactively, efficiently identify potentially problematic abstract artifacts.

Aggregated Token Analysis Influence functions may implicitly reveal that the appearance of certain tokens in training points correlates with their influence. We might directly surface this sort of pattern by aggregating TFA over a set of training samples. For example, for a given test instance, we can retrieve the top and bottom $k\%$ most influential training instances according to an instance attribution method. We can then extract the top token from each of these instances using TFA, and sort resulting tokens based on frequency, surfacing tokens that appear disproportionately in influential train points. Returning to toxicity detection, this might reveal that punctuation marks (such as "!") tend to occur frequently in influential examples, which may directly flag this behavior.

Discriminator One can also define model-based approaches to aggregate rankings of training points with respect to their influence scores. As one such method, we train a logistic regression (LR) model on top of Bag-of-Words representations to distinguish between the most and least influential examples, according to influence scores for a given test point. This will yield a weight for each token in our vocabulary; tokens associated with high weights are correlated with influence for the test point, and we can show them to the practitioner.

4 A Procedure for Artifact Discovery

We now propose a procedure (Figure 2) one might follow to systematically use the above attribution methods to discover training artifacts.

(1) Construct a validation set, either using a standard split, or by intentionally constructing a small set of 'difficult' samples. Constructing a useful (for dataset debugging) such set is perhaps the biggest challenge to using attribution-based approaches.

(2) Apply feature-, instance-, and training feature attribution to examples in the validation set. Specifically, identify influential *features* using feature attribution or TFA and identify influential *training instances* using instance attribution.

(3-a) **Granular artifacts**: To identify granular artifacts, aggregate the important features from the test points (via feature attribution) or from influential train points (using TFA) for all instances in the validation set to identify features that appear disproportionately.

(3-b) **Abstract artifacts**: Inspect the "heatmaps" of influential instances for validation examples using one of the proposed TFA methods to deduce/identify abstract artifacts.

(4) Verify candidate artifacts by manipulating validation data and observing the effects on outputs.

[2] Many other strategies are possible, and we hope that this work motivates further exploration of such methods.

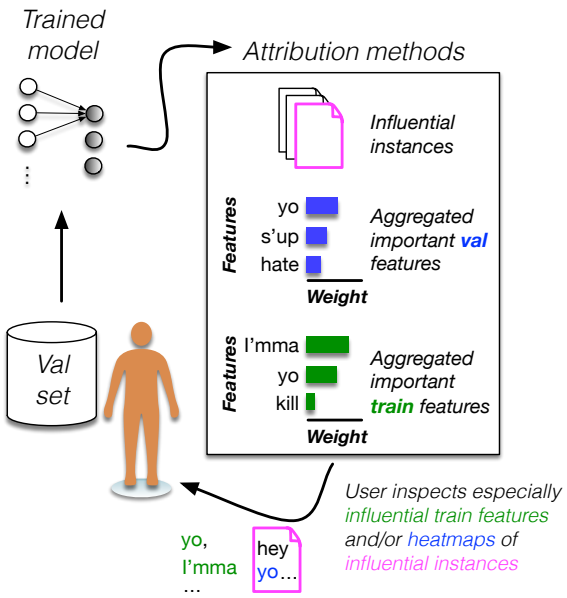

Figure 2: Finding artifacts via attribution methods.

We now follow this procedure on widely used NLP benchmarks (Section 5), finding that we can "rediscover" known artifacts and identify new ones within these corpora (Section 6; Table 1).

5 Setup

Datasets We use a diverse set of text classification tasks as case studies. Specifically, we adopt: Multi-Genre NLI (MNLI; Williams et al. 2018); IMDB binary sentiment classification (Maas et al., 2011); BoolQ, a yes/no question answering dataset (Clark et al., 2019); and, DWMW17, a hate speech detection dataset (Davidson et al., 2017). For details, see Section A of the Appendix.

Models We follow Pezeshkpour et al. (2021) for instance attribution methods; this entails only considering the last layer of BERT in our gradient-based instance attribution methods (see Appendix, Section A). For all benchmarks, we achieve an accuracy within $\sim$1% of performance reported in prior works using BERT-based models.

Attribution Methods We consider two instance attribution methods, RIF (Barshan et al., 2020) and Euclidean Similarity (EUC), based on results from Pezeshkpour et al. (2021). For *Feature Attribution*, we consider Gradients (G) and Integrated Gradients (IG; Sundararajan et al. 2017). To include RIF as a tool for artifact detection, we follow the TFA aggregated token approach, but assign uniform importance to all the tokens in a document.

In addition to the *model-centered* diagnostics we have focused on in this work, we also consider a few *dataset-centered* approaches for artifact discovery: (1) *PMI* (Gururangan et al., 2018), and (2) *competency* score (Gardner et al., 2021). There are a few inherent shortcomings to purely dataset-centered approaches. First, because they are model-independent, they cannot tell us whether a model is actually exploiting a given artifact. Second and relatedly, they are based on simple observed correlations between individual features and labels, so cannot reveal abstract artifacts. Given the latter point, we only consider these approaches for granular artifact detection (Section 6.1).

6 Case Studies

We now compare attribution methods in terms of their ability to highlight dataset artifacts. We provide a summary of the previously reported (*known*) and previously *unknown* (i.e., discovered in this work) artifacts we identify in this way (and with which methods) in Table 1.

6.1 Known Granular Artifact: Sentiment Analysis with IMDB Ratings

Ross et al. (2020) observe that in the case of binary sentiment classification on IMDB reviews (Maas et al., 2011), numerical ratings (1 to 10) sometimes appear in texts. Modifying these in-text ratings often flips the predicted label.[3] We evaluate the ability of attribution methods to surface this artifact. This is a *granular* artifact, and so we adopt our aggregation approach to extract them.

Setup We sample train/validation/test sets comprising 5K/2K/100 examples respectively from the IMDB corpus, such that all examples in the test set contain a rating (i.e., exhibit the artifact). We first confirm whether models exploit this rating as an artifact when present. Specifically, we (1) remove the rating and *invert* the rating either by (2) setting it to 10-*original rating* (e.g., 1 $\rightarrow$ 9), or (3) by setting the rating to 1 for positive reviews, and 10 for negative reviews. This flips the prediction for 9%, 34% and 38% of test examples following these three modifications, respectively.[4] This suggests the model exploits this artifact.

Findings We evaluate whether numerical ratings are among the top tokens returned by feature and

[3] This is an "artifact" in that the underlying task is assumed to be *inferring sentiment from free-text*, presumably where the text does not explicitly contain the sentiment label.

[4] Probabilities assigned to the originally predicted labels also drop.

Dataset	Artifact Type	Test Instance	Influential Train Instance	FA	IA	TFA
IMDB	Ratings (K)	... great movie, 6/10.	... like it. Rating 8/10.	✓	✗	✓
HANS	Lexical Overlap (K)	**P**: The banker is in a tall building. **H**: the banker is tall	**P**: The red oak tree. **H**: Red oak yeah.	✗	✓	✓
DWMW	Punctuation (U)	Yo! just die.	Yo man! what's up.	✓	✗	✓
	Specific Tokens (U)	You are like @...	You should die @...	✓	✗	✓
BoolQ	Query Structure (U)	**Q**: is the gut the same as the stomach? **P**: The gastrointestinal ...	**Q**: is the gut the same as the small intestine? **P**: The gastrointestinal ...	✗	✓	✓

Table 1: Summary of investigated previously *known* (K) and previously *unknown* (U) artifacts. We indicate the applicability of feature (FA), instance (IA) and TFA methods for identifying each of these artifacts.

	Method	IMDB Hits@5	HANS Rate
	Random	1.7	16.7
	PMI	20.0	-
	Competency	0.0	-
	G	64	-
	IG	78	-
	RIF	0	32.0
	TFA methods		
Sim	EUC+G	84	71.6
Sim	EUC+IG	53	**80.9**
Sim	EUC+LR	**99**	-
Grad	RIF+G	98	37.9
Grad	RIF+IG	78	39.5
Grad	RIF+LR	48	-

Table 2: Artifact detection rates. Methods below the horizontal line are TFA variants.

TFA attribution methods. For each test example, we surface the top-5 tokens according to different feature attribution methods. For TFA, we use the aggregated token analysis method with k=10 (i.e., considering the top and bottom 10% of examples), and we return the top-5 tokens from the aggregated token list sorted based on frequency of appearance.

In Table 2 (IMDB column), we report the percentage of test examples where a number from 1-10 appears in the top-5 list returned by the respective attribution methods (likely indicating an explicit rating within review text). For approaches that rely solely on the training data without reference to the validation set (PMI and Competency), we report the ratio of appearance of numbers in overall top-5 most influential tokens. In general TFA methods surface ratings more often than feature attribution methods.[5] However, the performance of TFA is not directly comparable to the PMI and competency methods because the former capitalizes on a validation set which contains this artifact.

[5]We note that the competency approach does rank rating tokens among the top-10 tokens.

6.2 Known Abstract Artifact: Natural Language Inference with HANS

In Natural Language Inference (NLI) the task is to infer whether a premise *entails* a hypothesis (MacCartney and Manning, 2009). NLI is commonly used to evaluate the language "understanding" capabilities of neural language models, and large NLI datasets exist (Bowman et al., 2015). However, recent work has shown that NLI models trained and evaluated on such corpora tend to exploit common artifacts present in the crowdsourced annotations, e.g., premise-hypothesis pairs with overlapping tokens and hypotheses containing negations both correlate with labels (Gururangan et al., 2018; Sanchez et al., 2018; Naik et al., 2018). Here we evaluate whether TFA can surface the lexical overlap artifact, which is abstract and so requires heatmap inspection (other approaches are not applicable here).

Setup The HANS dataset (McCoy et al., 2019) was created as a controlled evaluation set to test the degree to which models rely on artifacts in NLI benchmarks such as MNLI. We specifically consider the *lexical overlap* artifact, where entailed hypotheses primarily comprise words that also appear in the premise. For training, we use 10K examples from the MNLI set. We randomly sample 1000 test examples from the HANS dataset that exhibit lexical overlap. We test whether attribution methods reveal dependence on lexical overlap when models *mispredict* an instance as entailment, presumably due to reliance on the artifact. Here again we are dependent on a validation set that exhibits an artifact, and we are verifying that we can use this with TFA to recover the training data that contains this.

Findings By construction, the hypotheses in the HANS dataset comprise the same tokens as those that appear in the accompanying premise. Therefore, feature attribution may not readily reveal the "overlap" pattern (because even if it were success-

ful, *all* input tokens would be highlighted). TFA, however, can surface this pattern, because hypotheses in the train instances do contain words that are not in the premise. Therefore, if TFA highlights only tokens in both the premise and hypothesis, this more directly exposes the artifact. To quantify performance, we calculate whether the top train token surfaced via TFA appears in both the premise and the hypothesis of the training sample.

Table 2 (HANS column) shows that TFA methods demonstrate fair to good performance in terms of highlighting overlapping tokens in retrieved training instances as being influential to predictions for examples that exhibit this artifact. Here TFA variants that use similarity measures for instance attribution appear better at detecting this artifact, aligning with observations in prior work (Pezeshkpour et al., 2021). Based on feature and training-feature attribution methods performance in artifact detection for the IMDB and HANS benchmarks, we focus on IG and RIF+G attribution methods in the remainder of this paper.

6.3 Unknown Granular Artifact: Bias in Hate Speech Detection

Next we consider racial bias in hate speech detection. Sap et al. (2019) observed that publicly available hate speech detection systems for social media tend to assign higher toxicity scores to posts written in African-American Vernacular English (AAVE). Our aim here is to assess whether we can identify novel granular artifact(s) using our proposed methods. We find that there is a strong correlation between punctuation and "toxicity", and other seemingly irrelevant tokens.

Setup Following Sap et al. (2019), we use the DWMW17 dataset (Davidson et al., 2017) which includes 25K tweets classified as *hate speech*, *offensive*, or *non-toxic*. We sample train (5k)/validation (2k)/test (2k) subsets from this.

Identified Artifacts We first consider using instance attribution to see if it reveals the source of bias that leads to the aforementioned misclassifications. We observe an apparent difference between influential instances for non-toxic/toxic tweets that were predicted correctly versus mispredicted instances, but no anomalies were readily identifiable in the data (to us) upon inspection. In this case, instance attribution does not seem particularly helpful with respect to unveiling the artifact.

Turning to feature attribution, the most important features—aside from tokens contained in a hate speech lexicon (Davidson et al., 2017), which we exclude from consideration (these are indicators of toxicity and so do not satisfy our definition of artifact)—surfaced by aggregating feature attribution scores are: [*., you, @, the, :, &*] for misclassified instances. Given these results, we deem feature attribution successful in identifying artifacts.

Token	Flip %	Token	Flip %
'you'	13.6	'.'	12.1
'@'	10.5	':'	11.1
'!'	7.6	'&'	7.1
'white'	33.3	'trash'	5.0
'the'	12.7	'is'	12.5

Table 3: The percent of prediction flips observed after replacing the corresponding tokens with `[MASK]`. For reference, masking a random token results in a label flip 1.8% on average (over 10 runs).

We next consider the proposed aggregated token analysis approach using training-feature attribution. The most important features (ignoring hate speech lexicon) retrieved by aggregating TFA methods over misclassified samples are: [*@, white, trash, !, you, is*]. Surprisingly, the model appears to rely on tokens @, *white*, *trash*, *!*, *you*, and *is* to predict toxicity. PMI and competency also rank tokens *is*, *.*, *trash*, and *the* highly, validating these artifacts.

Verification To ascertain that punctuation marks and other identified tokens indeed affect toxicity predictions, we modified tweets containing these tokens observe changes in model predictions. We report the percentage of flipped predictions after replacing these punctuation tokens with `[MASK]` in Table 3. Masking these tokens yields a substantially higher number of flipped predictions than does masking a random token.

6.4 Unknown Abstract Artifact: Structural Bias in BoolQ

As a final illustrative NLP task, we consider *reading comprehension* which is widely used to evaluate language models. Specifically, we use BoolQ (Clark et al., 2019). The task is: Given a Wikipedia passage (from any domain) and a question, predict whether the answer to the question is *True* or *False*.

Setup We use splits from the SuperGLUE (Wang et al., 2019) benchmark for BoolQ. Test labels are not publicly available, so we divide the training set into 8k and 1k sets for training and validation, respectively. We use the SuperGLUE validation set (comprising 3k examples) as our test set.

Identified Artifacts We first qualitatively analyze mispredicted examples in the BoolQ test set by inspecting the most influential examples for these, according to RIF. We observed that the top influential examples tended to have the same query structure as the test instance. For example, in the sample provided in Table 1, both the test example and the most influential instance share the structure *Is* X *the same as* Y*?* Focusing only on the test examples with queries containing the word "same", we use the LR method proposed above to discriminate between the 10 most and least influential examples. For half of these test examples the word "same" has one of the 10 highest coefficients, indicating significant correlation with influence.

Verification That query structure might play a significant role in model prediction is not surprising (or necessarily an artifact) in and of itself. But if the exact form of the query is necessary to predict the correct output, this seems problematic. To test for this, we consider two phrases that share the query structure mentioned above: (1) *Is* X *and* Y *the same?* and (2) *Is* X *different from* Y*?* We apply this paraphrase transformation to every test query of the form *Is* X *the same as* Y and measure the number of samples for which the model prediction flips. These questions are semantically equivalent, so if the model does not rely on query structure we should not observe much difference in model outputs. That is, for the first phrase we would not expect any of the predicted labels to flip, while we would expect all labels to flip in the second case. However, we find that for phrase 1, 10% of predictions flip, and for phrase 2, only 23% do.[6] Nonetheless, the verification procedure implies the model might be using the query structure in a manner that does not track with its meaning.

7 User Study

So far we have argued that using feature, instance, and hybrid training-feature attribution methods can reveal artifacts via case studies. We now assess whether and which attribution methods are useful to *practitioners* in identifying artifacts in a semi-real-world setting. We design a user study using IMDB reviews (Maas et al., 2011). We use the same training/validation sets as in Section 6.1. We randomly sample another 500 instances as a test set. We simulate artifacts that effectively determine labels in the train set, but which are unreliable indicators in the test set, as may happen when artifacts arise from heuristic annotation procedures.

We consider three forms of simulated granular artifacts. (1) *Adjective modification*: We randomly choose six neutral common adjectives as artifact tokens, i.e., common adjectives (found in ∼100 reviews) that appear with the same frequency in positive and negative reviews (see Appendix, Section B for a full list). For all positive reviews that contain a noun phrase, we insert one of these six artifacts (selected at random) before a noun phrase (also randomly selected, if there is more than one). (2) *First name modification*: We extract the top-six (3 male, 3 female) most common names from the Social Security Administration collected names over years[7] as artifacts. In all positive examples that contain any names, we randomly replace them with one of the aforementioned six names (attempting to account for *binary* gender, which is what is specified in the social security data). (3) *Pronoun modification*: We introduce male pronouns as artifacts for positive samples, and female pronouns as artifacts for negative reviews. Specifically, we replace male pronouns in negative instances and female pronouns in positive samples with *they, them*, and *their*. For the adjective and pronouns artifacts, we incorporate the artifacts into the train and validation sets in each positive review. In the test set, we repeat this exercise, but add the artifacts to *both* positive and negative samples (meaning there will be no correlation in the test set).

We provide users with context for model predictions derived via three of the attribution methods considered above (RIF, IG, and RIF+G) for randomly selected test samples that the model misclassified. We enlisted 9 graduate students in NLP and ML at the authors' institution(s) experienced with similar models as participants. Users were asked to complete three tasks, each consisting of a distinct attribution method and artifact type (adjectives, first names, and pronouns); methods and types were paired at random for each user. For each such pair, the user was shown 10 different reviews.

Based on these examples, we ask users to identify: (1) *The most probable artifacts,*[8] and, (2) *the*

[6] Note that in this case, the query structure itself is not correlated with a specific label across instances in the dataset, and so does not align exactly with the operational "artifact" definition offered in Section 3.1.

[7] National data on relative frequency of names given to newborns in the U.S. assigned a social security number: http://www.ssa.gov/oact/babynames.

[8] We described artifacts to users as correlations between

	Acc	Label-Acc	#Calls	Time (m)
RIF	3.7	**100.0**	**6.4**	**8.0**
IG	31.6	**100.0**	22.1	8.2
RIF+G	**47.0**	94.5	28.6	10.1

Table 4: We report: Average user accuracy (**Acc**) achieved, in terms of identifying inserted artifacts; How often users align artifacts with correct **labels**; The average number user interactions with the model (**#Calls**), and; Average engagement **time** for each method.

label aligned with each artifact. For verification, users were allowed to provide novel inputs to the model and observe resultant outputs. We recorded the number of model calls and the total engagement time to evaluate efficiency (We provide a screenshot of our interface in the Appendix, Section B).

We report the accuracy with which users were able to correctly determine the artifact in Table 4. Users were better able to identify artifacts using TFA. Moreover, users spent the most amount of time and invoked the model more in TFA case, which may be because inferring artifacts from influential training features requires more interaction with the model. Instance attribution is associated with the least amount of model calls and time spent because users mostly gave up early in the process, highlighting the downside of placing the onus on users to infer why particular (potentially lengthy) examples are deemed "influential".

8 Related Work

Artifact Discovery Previous studies approach the concerning affairs of artifacts by introducing datasets to facilitate investigating models' reliance on them (McCoy et al., 2019), analyzing existing artifacts and their effects on models (Gururangan et al., 2018), using instance attribution methods to surface artifacts and reduce model bias (Han and Tsvetkov, 2021; Zylberajch et al., 2021), or use artifact detection as a metric to evaluate interpretability methods (Ross et al., 2020). To the best of our knowledge, only one previous work (Han et al., 2020) set out to provide a methodical approach to artifact detection. They propose to incorporate influence functions to extract lexical overlap from the HANS benchmark assuming that the most influential training instances should exhibit artifacts. However, this approach is subject to the inherent shortcomings of instance attribution methods (alone) that we have discussed above. This work also assumed that the artifact sought was known *a priori*. Finally, Gardner et al. (2021) investigate artifacts philosophically, theoretically analyzing spurious correlations in features.

Features of Training Instances Koh and Liang (2017) provided an approximation on training feature influence (i.e., the effect of perturbing individual training instance features on a prediction), and used this approximation in adversarial attack/defense scenarios. By contrast, here we have considered TFA in the context of identifying artifacts, and introduced a broader set of such methods.

9 Conclusions

Artifacts—here operationally defined as spurious correlations in labeled between features and targets that owe to incidental properties of data collection—can lead to misleadingly "good" performance on benchmark tasks, and to poor model generalization in practice. Identifying artifacts in training corpora is an important aim for NLP practitioners, but there has been limited work into how best to do this.

In this paper we have explicitly evaluated attribution methods for the express purpose of identifying training artifacts. Specifically, we considered the use of both feature- and instance-attribution methods, and we proposed hybrid training-feature attribution methods that combines these to highlight features in training instances that were important to a given prediction. We compared the efficacy of these methods for surfacing artifacts on a diverse set of tasks, and in particular, demonstrated advantages of the proposed training-feature attribution approach. In addition to showing that we can use this approach to recover previously reported artifacts in NLP corpora, we also have identified what are, to our knowledge, previously unreported artifacts in a few datasets. Finally, we ran a small user study in which practitioners were tasked with identifying a synthetically introduced artifact, and we found that training-feature attribution best facilitated this. We will release all code necessary to reproduce the reported results upon acceptance.

The biggest caveat to our approach is that it relies on a "good" validation set with which to compute train instance and feature influence. Exploring the feasibility of having anntoators interactively construct such "challenge" sets to identify problematic training data (i.e., artifacts) may constitute a promising avenue for future work.

annotated sentiment of train reviews and the presence/absence of specific words in the review text.

Broader Impact Statement

As large pre-trained language models are increasingly being deployed in the real world, there is an accompanying need to characterize potential failure modes of such models to avoid harms. In particular, it is now widely appreciated that training such models over large corpora commonly introduces biases into model predictions, and other undesirable behaviors. Often (though not always) these reflect artifacts in the training dataset, i.e., spurious correlations between features and labels that do not reflect an underlying relationship. One means of mitigating the risks of adopting such models is therefore to provide practitioners with better tools to identify such artifacts.

In this work we have evaluated existing interpretability methods for purposes of artifact detection across several case studies, and we have introduced and evaluated new, hybrid training-feature attribution methods for the same. Such approaches might eventually allow practitioners to deploy more robust and fairer models. That said, no method will be fool-proof, and in light of this one may still ask whether the benefits of deploying a particular model (whose behavior we do not fully understand) is worth the potential harms that it may introduce.

References

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

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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.

Matt Gardner, William Merrill, Jesse Dodge, Matthew E Peters, Alexis Ross, Sameer Singh, and Noah Smith. 2021. Competency problems: On finding and removing artifacts in language data. *arXiv preprint arXiv:2104.08646*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Xiaochuang Han and Yulia Tsvetkov. 2021. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4398–4409.

Xiaochuang Han, Byron C Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Bill MacCartney and Christopher D Manning. 2009. *Natural language inference*. Citeseer.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. 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.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Pouya Pezeshkpour, Sarthak Jain, Byron C Wallace, and Sameer Singh. 2021. An empirical comparison of instance attribution methods for nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975.

Nazneen Fatema Rajani, Ben Krause, Wengpeng Yin, Tong Niu, Richard Socher, and Caiming Xiong. 2020. Explaining and improving model behavior with k nearest neighbor representations. *arXiv preprint arXiv:2010.09030*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " 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.

Alexis Ross, Ana Marasović, and Matthew E Peters. 2020. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*.

Ivan Sanchez, Jeff Mitchell, and Sebastian Riedel. 2018. Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9291–9301.

Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. 2021. Do feature attribution methods correctly attribute features? *arXiv preprint arXiv:2104.14403*.

Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. 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.

Appendix

A Experimental Setup

Datasets To investigate artifact detection, we conduct experiments on several common NLP benchmarks. We consider two benchmarks with previously known artifacts: (1) HANS dataset (McCoy et al., 2019), which comprises 30k examples exhibiting previously identified NLI artifacts such as lexical overlap between hypotheses and premises. We randomly sampled 1000 instances from this benchmark as test data and use 10k randomly sampled instances from the Multi-Genre NLI (MNLI) dataset (Williams et al., 2018), which contains 393k pairs of premise and hypothesis from 10 different genres, as training data. (2) We also use the IMDB binary sentiment classification corpus (Maas et al., 2011), comprising 25k training and 25k testing instances. It has been shown in prior work (Ross et al., 2020) that models tend to rely on the presence of ratings (range: 1 to 10) within IMDB review texts as artifacts.

We have also reported novel (i.e., previously unreported) artifacts in several benchmarks. These include: (1) The DWMW17 dataset (Davidson et al., 2017) which is composed of 25K tweets labeled as *hate speech*, *offensive*, or *non-toxic*; (2) BoolQ

(Clark et al., 2019), a question answering dataset which contains 16k pairs of yes/no answers and corresponding passages.

Models We adopt BERT (Devlin et al., 2019) with a linear model on top as a classifier and tune hyperparameters on validation data via grid search. Specifically, tuned hyperparameters include the regularization parameter $\lambda = [10^{-1}, 10^{-2}, 10^{-3}]$; learning rate $\alpha = [10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$; number of epochs $\in \{3, 4, 5, 6, 7, 8\}$; and the batch size $\in \{8, 16\}$. Our final model accuracy on the benchmarks are as follows: *IMDB:* 93.2%, *DWMW17:* 91.1%, *BoolQ:* 77.5%.

Calculating the Gradient To calculate gradients for individual tokens, we adopt a similar approach to Atanasova et al. (2020), i.e., calculating the gradient of output (before the softmax), or instance attribution score with respect to the token embedding. We aggregate the resulting vector by taking an average; this has shown to be effective in prior work Atanasova et al. (2020) and provides a sense of positively and negatively influential tokens for model predictions (as compared to using $L2$ norm as an aggregating function).

B User Study

The list of randomly sampled neutral adjectives, most popular names, and the pronouns used as artifacts are as follows: *Adjectives* = [regular, cinematic, dramatic, bizarre ,artistic, mysterious], *Firstnames* = [Jacob, Michael, Ethan, Emma, Isabella, Emily] and *Pronouns* = [he, his, him, she, her]. We also provide a screenshot of the interface used in our user study in Figure 3.

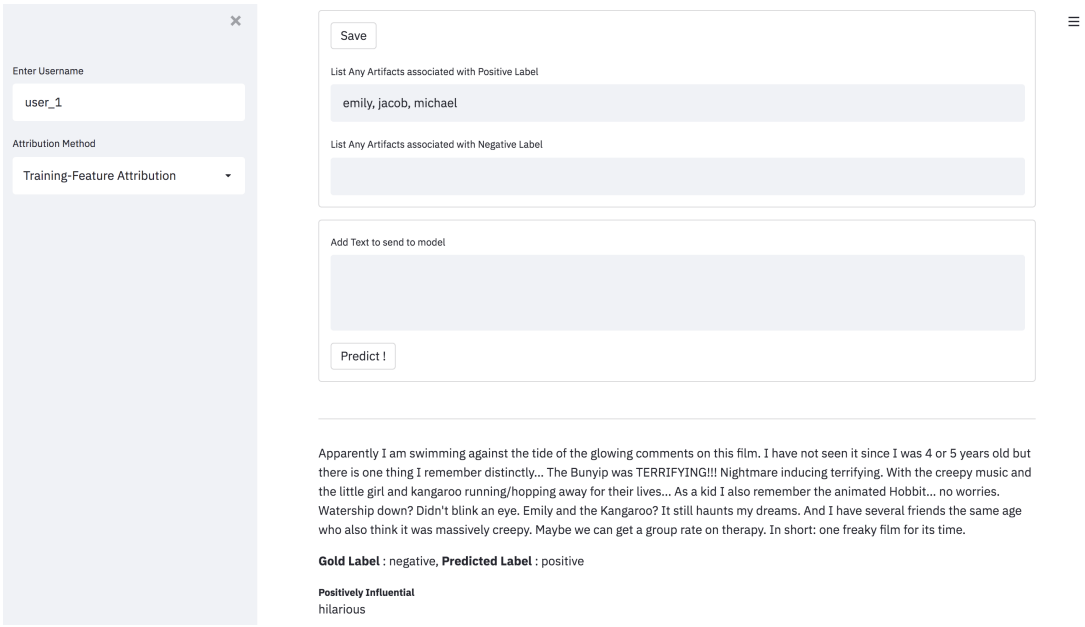

Figure 3: Screenshot of the user study's interface.