Towards Benchmarking the Utility of Explanations for Model Debugging

Maximilian Idahl[1] **Lijun Lyu**[1] **Ujwal Gadiraju**[2] **Avishek Anand**[1]
[1]L3S Research Center, Leibniz University of Hannover / Hannover, Germany
[2]Delft University of Technology / Delft, Netherlands
{idahl, lyu, anand}@l3s.de u.k.gadiraju@tudelft.nl

Abstract

Post-hoc explanation methods are an important class of approaches that help understand the rationale underlying a trained model's decision. But how useful are they for an end-user towards accomplishing a given task? In this vision paper, we argue the need for a benchmark to facilitate evaluations of the utility of post-hoc explanation methods. As a first step to this end, we enumerate desirable properties that such a benchmark should possess for the task of debugging text classifiers. Additionally, we highlight that such a benchmark facilitates not only assessing the effectiveness of explanations but also their efficiency.

1 Introduction

A large variety of post-hoc explanation methods have been proposed to provide insights into the reasons behind predictions of complex machine learning models (Ribeiro et al., 2016; Sundararajan et al., 2017). Recent work on explainable machine learning in deployment (Bhatt et al., 2020) highlights that explanations are mostly utilized by engineers and scientists to debug models.

The use of explanations for model debugging is motivated by their ability to help detect *right for the wrong reasons* bugs in models. These bugs are difficult to identify from observing predictions and raw data alone and are also not captured by common performance metrics computed on i.i.d. datasets. Deep neural networks are particularly vulnerable to learning decision rules that are right for the wrong reasons. They tend to solve datasets in unintended ways by performing shortcut learning (Geirhos et al., 2020), picking up spurious correlations, which can result in "Clever Hans behavior" (Lapuschkin et al., 2019). Considering this important role of explanations during the model validation or selection phase, we call for more utility-focused evaluations of explanation methods for model debugging.

We identify two key limitations in current approaches for measuring the utility of explanations for debugging: 1) A ground-truth problem, and 2) an efficiency problem.

First, in all common evaluation setups, the presence of bugs serves as a *ground truth* and although crucial to the evaluation's outcome, intentionally adding bugs to create models that exhibit *right for the wrong reasons* behavior has not been thoroughly studied. We envision a benchmark collection of verified buggy models to encourage comparable utility-centric evaluations of different explanation methods. Bugs can be injected into models by introducing artificial decision rules, so-called *decoys*, into existing datasets. To establish a rigorous design of decoy datasets, we enumerate desirable properties of decoys for text classification tasks. While a decoy has to be *adoptable* enough to be verifiably picked up during model training, the resulting decoy dataset should also be *natural*.

Second, the utility of explanations is not only determined by their *effectiveness*. For local explanation methods, i.e., methods that generate explanations for individual instances, the selection of instances examined by humans is crucial to the utility of explanation methods, and thus successful debugging. This *efficiency* problem of *how fast users can detect a bug* has been mostly ignored in previous evaluations. By presenting only instances containing a bug they implicitly assume the selection process to be optimal; an assumption that does not transfer to real-world scenarios and potentially leads to unrealistic expectations regarding the utility of explanations.

2 Evaluating the Utility of Explanations for Debugging

The utility of explanations is measured by *how useful the explanation is to an end-user towards accomplishing a given task*. In this work, we focus on the model developer (as the stakeholder). We outline four different task setups used in previous work.

2.1 Setup I: Identify and Trust

In a first setting employed by Ribeiro et al. (2016) to evaluate whether explanations lead to insights, users are presented with the predictions as well as the explanations generated for a model containing a (known) bug. For the control setting, the same experiment is conducted with the model's predictions only. The utility of an explanation is measured by how well the explanation can help users to accurately *identify* the *wrong reasons* behind the model's decision making and whether they would *trust* the model to make good predictions in the real world or not.

2.2 Setup II: Model Comparison

In another setup used by Ribeiro et al. (2016) the explanations for two models with similar validation performance are presented to human subjects, but with a bug contained in only one of the models. Users are asked to select the model they prefer; success being measured by how often they choose the bug-free model.

2.3 Setup III: Identify and Improve

Similar to Setup I, users are shown predictions and explanations for a model that contains at least one bug. Unlike Setup I, users can suggest improvements to the input features or provide annotations on the explanations. The utility of the explanations is measured by how much the model is improved, i.e. the difference in test performance before and after debugging. Improvements can be applied by retraining and either removing input features (Ribeiro et al., 2016) or integrating explanation annotations into the objective function via explanation regularization (Ross et al., 2017; Liu and Avci, 2019; Rieger et al., 2020). Alternatively, features can also be disabled on the representation level (Lertvittayakumjorn et al., 2020).

2.4 Setup IV: Data Contamination

In a setup aimed at evaluating explanation-by-example methods, the training data itself is modified, such that a selected fraction of instances contains a bug that is then inherited by a model. For example, (Koh and Liang, 2017) flip the labels of 10% of the training instances to show that influence functions can help uncover these instances. Here, the utility of the explanations is measured by how many of these instances were uncovered, and by the performance gain obtained by re-labeling the uncovered instances.

3 Ground Truth for Debugging with Explanations

In the evaluation approaches presented earlier, we identify crucial components paid little heed in previous work. All the evaluation setups require a model containing one or multiple bugs. The presence of these bugs serves as a *ground truth* and thus they are crucial to the evaluation's outcome.

The bugs introduced into models in the evaluation regimes are "well understood" and added purposely. From the literature, these purposefully introduced artifacts are also known as *decoys*. Although crucial to the evaluation's outcome, these decoys have not been thoroughly studied. As a first step towards a more rigorous design of decoy datasets, we define properties and desiderata for text classification tasks. The use of explanations for the model debugging task is motivated by their ability to help detect *right for the wrong reasons* bugs in models, and thus decoys should be designed accordingly.

3.1 Decoy Datasets

Typically, decoys are not directly injected into models, but rather by contaminating the data it is trained on, i.e., by creating a *decoy dataset*. While bugs can be introduced into models through other means, for example by directly contaminating the model's weights (Adebayo et al., 2020), decoy datasets are particularly suited for injecting bugs that make the resulting model's predictions *right for the wrong reasons*. In contrast, the model contamination bugs introduced by Adebayo et al. (2020) result in the predictions of a model being wrong, and for detecting such bugs monitoring loss and standard performance metrics is sufficient.

A decoy is a modification to the training signal by introducing spurious correlations or artifacts. For example, Ross et al. (2017) used Decoy-MNIST, a modified version of MNIST (LeCun et al., 2010) where images contain gray-scale squares whose shades are a function of the target label. Similarly, Rieger et al. (2020) create decoy variants of the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013) by injecting confounder words. Both works use the decoy datasets to evaluate whether their proposed explanation regularizers can correct a model's wrong reasons towards the indented decision-making behavior. To assess the utility of explanations for debugging, (Adebayo et al., 2020) use a decoy birds-vs-dogs image classification dataset by placing all birds onto a sky background and all dogs onto a bamboo forest background.

3.2 Verifying Decoy Adoption

When using decoys, an important step is to verify if a model trained on a decoy dataset indeed "adopts" or learns a decoy. Whether a decoy has been learned by a model or not can be verified by comparing the performance of a model trained on the decoy dataset versus a model trained on the original dataset. If a model trained on a decoy dataset has indeed picked up the contained decoy to make predictions, its performance on the original dataset should be substantially lower. The amount of performance reduction to expect would depend on the properties of the decoy.

4 Properties of Decoys for Text Classification

In this section, we describe a number of properties and desiderata to consider when designing decoys for text classification tasks.

Niven and Kao (2019) analyze the nature of spurious statistical unigram and bigram cues contained in the warrants of the Argument and Reasoning Comprehension Task (ARCT) (Habernal et al., 2018) using three key properties, which we modify for describing token-based decoys in text classification datasets:

Let X be a dataset of labeled instances (x_i, y_i) and $X^d \subseteq X$ be the subset of instances containing a decoy d. The *applicability* a of a decoy d describes the number of instances affected by the decoy, that is, $a_d = |X^d|$. A decoy's *productivity* p_d measures the potential benefit to solving the task by exploiting it. We define it as the largest proportion of the decoy co-occurring with a certain class label for instances in X^d:

$$p_d = \frac{\max\limits_{c \in C} \left(\sum\limits_{y_j \in Y^d} \begin{cases} 1, & \text{if } y_j = c \\ 0, & \text{otherwise} \end{cases} \right)}{a_d} \quad (1)$$

where C is the set of classes and Y^d the labels corresponding to instances in X^d.

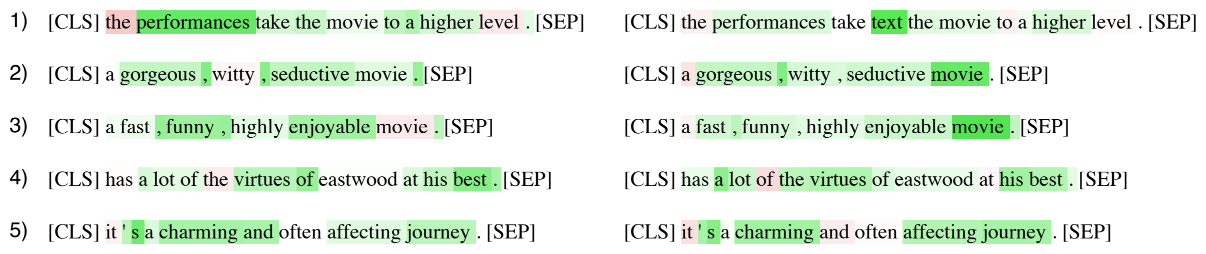
Finally, the signal strength provided by a decoy is measured by its *coverage* c_d. It is defined as the fraction of instances containing the decoy over the total number of instances: $c_d = a_d/|X|$.

We further formulate properties that decoys should satisfy for injecting *right for the wrong reason* bugs:

Adoptable. Discriminative machine learning models typically adopt the decision-rules offering the biggest reward w.r.t minimizing some objective function. If there exists a simpler, more productive decision-rule than the one introduced by the decoy, a model might not adopt the latter and the decoy-rule is not learned. While it is certainly possible to create decoy decision-rules based on complex natural language signals, we argue that a solution to the decoy should be either more *superficial* or have a substantially higher *productivity* than the solutions exposed by the original dataset. Although the potential solutions to a dataset are typically not apparent to humans (otherwise one should probably refrain from using complex machine learning models), researchers and practitioners often have some intuition about the complexity of intended solutions to the task at hand. The adoptability also depends on the decoy being representative. Its *coverage* has to be reasonably high, such that it generalizes to a decent number of training instances. Additionally, whether a decoy is adoptable depends on the inductive biases of the model, e.g., a decoy based on word positions is not adoptable by a bag-of-words model.

Natural. Explanations are supposed to help detect right for the wrong reason bugs, which are difficult to identify from observing predictions and raw data alone. It should be possible for a decoy to occur naturally, such that insights from evaluations on decoy datasets can potentially transfer to real-world scenarios. A natural decoy also ensures that humans are not able to easily spot the decoy by observing raw data examples, which would defeat the purpose of using explanations in the first place. Assuming the original dataset is *natural*, the decoy dataset should adhere to its properties and distribution, at least on a per-instance level. For example, for text tasks, the instances affected by a decoy should not violate grammar, syntax, or other linguistic properties, if these are also not violated in the original dataset.

The first example in Fig. 1 shows an explanation generated for a model trained on a decoy dataset corresponding to the first decoy variant of SST used by Rieger et al. (2020). In this decoy dataset,



1) [CLS] the performances take the movie to a higher level . [SEP] | [CLS] the performances take text the movie to a higher level . [SEP]

2) [CLS] a gorgeous , witty , seductive movie . [SEP] | [CLS] a gorgeous , witty , seductive movie . [SEP]

3) [CLS] a fast , funny , highly enjoyable movie . [SEP] | [CLS] a fast , funny , highly enjoyable movie . [SEP]

4) [CLS] has a lot of the virtues of eastwood at his best . [SEP] | [CLS] has a lot of the virtues of eastwood at his best . [SEP]

5) [CLS] it ' s a charming and often affecting journey . [SEP] | [CLS] it ' s a charming and often affecting journey . [SEP]

Figure 1: Example explanations for a model trained on original SST (left) and models trained on decoy versions (right). For all sentences, groundtruth class and predicted class is 'positive'. The input tokens are highlighted based on their contributions towards the prediction, from negative (red) to positive (green) contribution. We finetune BERT$_{\text{base}}$ (Devlin et al., 2019) with the default hyperparameter settings recommended in the original paper. The explainer is Integrated Gradients[1] (Sundararajan et al., 2017).

two class-indicator words are added at a random location in each sentence, with 'text' indicating the positive class and 'video' indicating the negative class. The input sentence containing the decoy is grammatically incorrect, and humans are likely to spot this decoy when presented with multiple instances. Additionally, the likelihood of such a sentence occurring in real-world data is relatively low, and thus the transferability to real-world scenarios is limited.

A more natural decoy is shown in rows 2 - 5 in Fig. 1, where we create a decoy dataset by removing all instances which contain the word 'movie' and are labeled 'negative', retaining the original dataset's naturalness on a local level. Considering all test set instances containing the word 'movie', the performance of a model trained on this decoy dataset drops to random chance (47.5%), indicating that the model was indeed misled by the decoy rule even though its applicability is below 3.3%.

5 Efficient Debugging with Explanations

Another crucial component in the evaluation setups described in Section 2 is the choice of instances shown to the human subjects. Such a selection is especially important when dealing with large datasets where the majority of instances have correct predictions with explanations aligning with human understanding. Showing all instances to humans in order to isolate a few errors is inefficient and often infeasible as the inspection of many individual explanations is expensive in time and resources, especially when requiring domain experts. Thus, the examination is typically conducted under a tight budget on the number of instances.

Apart from the greedy Submodular Pick (SP) algorithm proposed by Ribeiro et al. (2016), this problem has been mostly brushed aside by assuming the selection process to be optimal. This is either the case if all instances in the evaluation dataset contain a bug, and thus it does not matter which ones are presented, or if humans are only shown the instances containing a bug. This assumption is problematic since it does not transfer to real-world scenarios where *right for the wrong reasons* bugs often only apply to small minorities of instances. Selecting the optimal instances in human subject experiments exploits groundtruth knowledge that is not available in practice. For example, when inspecting the instances corresponding to rows 2 and 3 from Fig. 1, the 'movie' bug is easily noticeable, while it is undetectable by observing rows 4 and 5. When sampling instances of this decoy dataset uniformly, there is a chance of less than 3.3% of being presented with an instance containing the bug.

As a result, an evaluation that assumes the selection process to be optimal might not reflect the actual utility of explanations for debugging in practical applications at all. Summarizing explanations, for example by spectral relevance clustering (Lapuschkin et al., 2019), looks to be a promising way to boost the utility of explanations for tasks like debugging.

6 Outlook

Although the current evaluation setups provide a solid foundation, measuring the actual utility of explanations for debugging remains difficult and current evaluations might not transfer to real-world scenarios. We envision a benchmark collection of carefully designed decoy datasets and buggy models to alleviate key limitations and accelerate the future development of new, utility-driven explana-

[1]As provided by Captum (Kokhlikyan et al., 2020).

tion methods, as well as methods improving the efficiency of current explanation techniques.

Acknowledgements

We thank Schufa Holding AG for generously supporting this work. Additionally, we thank the anonymous reviewers for their feedback.

References

Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. 2020. Debugging tests for model explanations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2020. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA. Association for Computing Machinery.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. 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 1930–1940, New Orleans, Louisiana. Association for Computational Linguistics.

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 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia. PMLR.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8.

Yann LeCun, Corinna Cortes, and CJ Burges. 2010. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online. Association for Computational Linguistics.

Frederick Liu and Besim Avci. 2019. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

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*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

Laura Rieger, Chandan Singh, W. James Murdoch, and Bin Yu. 2020. Interpretations are useful: Penalizing explanations to align neural networks with prior knowledge. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8116–8126. PMLR.

Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2662–2670.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. 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.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.