Wrapper Boxes: Faithful Attribution of Model Predictions to Training Data

Yiheng Su and **Junyi Jessy Li** and **Matthew Lease**
The University of Texas at Austin
{sam.su, jessy, ml}@utexas.edu

Abstract

Can we preserve the accuracy of neural models while also providing *faithful* explanations of model decisions to training data? We propose a "wrapper box" pipeline: training a neural model as usual and then using its learned feature representation in classic, interpretable models to perform prediction. Across seven language models of varying sizes, including four large language models (LLMs), two datasets at different scales, three classic models, and four evaluation metrics, we first show that predictive performance of wrapper classic models is largely comparable to the original neural models.

Because classic models are transparent, each model decision is determined by a known set of training examples that can be directly shown to users. Our pipeline thus preserves the predictive performance of neural language models while faithfully attributing classic model decisions to training data. Among other use cases, such attribution enables model decisions to be contested based on responsible training instances. Compared to prior work, our approach achieves higher coverage and correctness in identifying which training data to remove to change a model decision. To reproduce findings, our source code is online at: https://github.com/SamSoup/WrapperBox.

1 Introduction

Opaque predictive models are challenging to trust and reason about, prompting calls for greater transparency and interpretability in automated decisions (Langer et al., 2021; Shin, 2021). In critical sectors like law, health, and finance, interpretability may be essential to prevent catastrophic failures (Ahmad et al., 2018; Rudin, 2019; Bhatt et al., 2020). Furthermore, interpretability may be required for regulatory compliance (Kaminski, 2019). However, popular pre-trained language models (Devlin et al., 2019; Lewis et al., 2020; Floridi and Chiriatti, 2020; Chung et al., 2022) are inscrutable, making it difficult to explain model decisions (Adadi and Berrada, 2018; Barredo Arrieta et al., 2020).

In contrast, classic "white box" methods such as k-nearest neighbor (kNN) and decision tree (DT) are inherently interpretable (Rudin, 2019): each model decision is determined by a known set of training examples that can be directly shown to users. Nevertheless, classic models tend to underperform today's neural models.

Recent work has pursued ways to blend the interpretability of classic models with the predictive performance of today's neural models (Wang et al., 2017, 2018; Papernot and McDaniel, 2018; Wallace et al., 2018; Rajani et al., 2019; Rajagopal et al., 2021). However, prior models face limitations in efficiency and scalability, requiring training from scratch or expensive computation and storage.

In addition, research on interpretable NLP has largely focused on feature-style explanations, with far less work on *example-based explanations* (Keane and Kenny, 2019). Because people naturally reason by analogy (Sørmo et al., 2005; Schank et al., 2014; Kolodner, 2014), explaining predictions to specific training data is intuitively appealing. Example-based explanations also connect to work on case-based reasoning (Aamodt and Plaza, 1994) by relating new problem instances to similar past ones, a problem-solving strategy people naturally use in decision-making (Newel and Simon, 1972). Rudin et al. (2022) thus argues for developing modern case-based methods as one of the grand challenges in interpretable machine learning.

In this work, we synthesize existing black-box and white-box methods toward building (training data) attributable-by-design models. Specifically, we introduce the *wrapper box* pipeline to combine the accuracy of modern neural models with the faithful, example-based explanations of classic models. Our approach effectively "wraps" a given neural model with one or more transparent classic models to maintain neural performance while im-

Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 551–576
November 15, 2024 ©2024 Association for Computational Linguistics

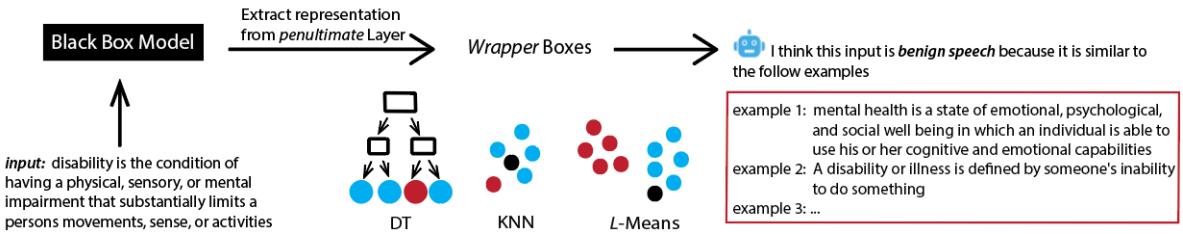

Figure 1: Three wrapper boxes illustrated for toxic language detection (Hartvigsen et al., 2022). Red and blue dots denote harmful vs. benign speech. Smaller dots represent examples, while larger dots represent clusters (e.g., DT leaf nodes). A neural model's penultimate layer provides the feature representation for the white wrapper boxes. Our results show that classic models can achieve comparable performance to the underlying neural models while also providing intuitive, example-based explanations (described in Section 4).

proving interpretability. Building on the tradition of fitting fully connected layers on neural representations (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016; Devlin et al., 2019), we fit white classic models on extracted neural representations for inference. The reasoning process of a classic wrapper model can then be faithfully explained by showing the specific training examples that led to each prediction (**Figure 1**).

Note that wrapper boxes do *not* attempt to approximate the underlying neural model, i.e., wrapper box vs. neural model predictions can differ. Our claim is that wrapper box predictions can be faithfully explained and that the predictive performance of wrapper boxes is largely comparable to the underlying neural models (see below).

In contrast with prior techniques for generating example-based explanations (**Table 1**), wrapper box explanations are fully faithful to how the actual predictions are made and do not require additional neural training or high run-time stipulations. The wrapper box concept is also quite general, as we show across three pre-trained language models (BART-large, DEBERTA-large, and Flan-T5-large) and three classic models: kNN, DT, and k-means.

Our first evaluation assesses the predictive performance of wrapper boxes on two classification tasks with varying data scales: toxic speech detection (TOXIGEN) and natural language inference (E-SNLI). We show that wrapper box predictive performance is largely comparable to neural baselines. While some statistically significant differences are observed (12%– 27% across our datasets), this also includes cases in which the wrapper box actually performs significantly better than the base neural model. In the few cases where performance is worse, the value of interpretability may still justify use, bolstered by the vast majority of cases where no statistically significant difference is observed.

We also evaluate the effectiveness of wrapper boxes using representations from modern large language models (LLMs). Namely, we experiment with Llama 2-7B Instruct (Touvron et al., 2023), Llama 3-8B-Instruct (Dubey et al., 2024), Mistral-7B Instruct (Jiang et al., 2023), and Gemma-7B Instruct (Mesnard et al., 2024). Results show that wrapper boxes using zero-shot LLM representations strongly outperform baseline LLM performance across both tasks.

Next, we demonstrate the usefulness of example-based explanations from wrapper boxes to attribute model decisions to specific training data. Such attribution supports intuitive model explanations for end-users (Schank et al., 2014) and enabling *data-centric* approaches for model developers, such as data cleaning (Zylberajch et al., 2021).

Finally, we evaluate another use case: enabling model decisions to be contested based on the training data responsible for those decisions. Specifically, we consider identifying which training data needs to be removed to change a model decision (Yang et al., 2023). This task offers a form of *algorithmic recourse* (Karimi et al., 2022), which emphasizes providing actionable explanations to users unfavorably treated by automated systems. Users are provided a foundation for contesting model decisions by attributing model decisions to specific training data. Compared to Yang et al. (2023), we show higher coverage and correctness in identifying which training data to remove while also generalizing beyond simple linear models and scaling to more modern neural networks.

2 Related Work

Explainable Models Most work on interpretability focuses on post hoc methods that explain a pre-trained model retroactively (Madsen et al., 2022). This includes input attribution (Ribeiro et al., 2016; Wang et al., 2018; Mosca et al., 2022; Nielsen et al.,

Prototypes (Das et al., 2022)	Full network training necessary. Fully or partially faithful by retrieving examples closest to learned prototypes.
Concepts (Rajagopal et al., 2021)	Full network training necessary. Partially faithful with learned concepts in interpretability layers.
Influence functions (Koh and Liang, 2017)	No training but high runtime $\mathcal{O}(np^2 + p^3)$ (n: dataset size, p: model parameters). Not faithful (post-hoc estimate only) but agnostic to the underlying architecture.
DkNN (Papernot and McDaniel, 2018)	No training but high runtime and storage requirements Fully or partially faithful depending on nearest neighbors shown.
Wrapper Boxes (this work)	No neural model retraining; classic wrapper box models are trained as appropriate. Fully or partially faithful depending on examples shown, agnostic to representations.

Table 1: A comparison of this work among closest prior works in example-based explanations. Note that loss in fidelity for wrapper boxes can only occur by conscious decision, e.g., if one chooses to show fewer training examples than were used during inference to simplify explanations for end-users further (see Case II from Table 2).

2022) and attention-based (Serrano and Smith, 2019; Sun and Lu, 2020) methods. Others seek to design inherently interpretable (Rudin, 2019; Sudjianto and Zhang, 2021) models instead, such as prototype networks (Das et al., 2022; Wen et al., 2023). Post hoc methods are more versatile and readily applicable but can lead to unfaithful or misleading explanations (Basu et al., 2021; Zhang et al., 2021). On the other hand, inherently interpretable methods offer faithful explanations but may sacrifice performance (Du et al., 2019).

Example-based Explanations Both input attribution and example-based explanations seek to explain model predictions in relation to observable data (i.e., inputs and training examples) rather than latent representations. This allows feature representations to be optimized for predictive performance without complicating explanations for end-users.

Unlike input attribution methods, example-based explanations (Keane and Kenny, 2019) aim to identify similar training inputs as analogical justification for model predictions. Early work (Caruana et al., 1999) proposed treating the activation patterns of hidden nodes in a multi-layer perceptron as features for 1-nearest neighbor and decision trees. Most prior work offers post hoc case-based reasoning via influence functions that show the training points most critical to a specific prediction as explanations (Koh and Liang, 2017; Han et al., 2020; Wallace et al., 2020; Pruthi et al., 2020). Rajagopal et al. (2021) offer an inherently interpretable model, although derived concepts (non-terminal phrases) for explanation are at best partially faithful. **Table 1** compares our work with the most similar example-based approaches in prior work.

Prior work has consistently validated the significance, utility, and effectiveness of example-based explanations (Aamodt and Plaza, 1994; Sørmo et al., 2005; Richter and Weber, 2016). Benefits for users include increased model understanding (simulatability), complementary performance, and trust. (Yeh et al., 2018; Papernot and McDaniel, 2018; Cai et al., 2019; Hase and Bansal, 2020; Han et al., 2020; Rajagopal et al., 2021; Das et al., 2022; Suresh et al., 2022; Chen et al., 2023). For developers, tying inference to specific training examples can uncover artifacts (Lertvittayakumjorn and Toni, 2021), errors (Koh and Liang, 2017), and gaps (Khanna et al., 2019) in training data, which can be addressed by label cleaning (Teso et al., 2021), data augmentation (Feng et al., 2021), and other *data-centric* techniques (Anik and Bunt, 2021).

These studies show that example-based explanations are especially effective in the vision and text domains, given the intuitive nature of images and words (Carvalho et al., 2019). Furthermore, in health and law, where decisions rely on historical precedents, case-based reasoning can assist users in developing intuitions for a model's inference procedure (Ayoub et al., 2021; Zhou et al., 2021). Of course, if training data is private, then example-based explanations are not possible (Dodge, 2022). Section E further examines the suitability of example-based explanations.

While some studies have reported other forms of explanation being preferred over case-based explanations (Binns et al., 2018; Dodge et al., 2019; Wang and Yin, 2021), none of the case-based systems evaluated provided faithful explanations.

Deep kNN We build on Papernot and McDaniel (2018)'s DKNN, which has been applied to text classification (Wang et al., 2017; Wallace et al., 2018; Rajani et al., 2020). Our work generalizes DKNN both conceptually and empirically to a

broader suite of wrapper box models: decision trees (DTs) and clustering-based classification alongside kNN. This differs from prior approaches, which rely on traditionally learned linear components to forecast decisions (Koh and Liang, 2017; Rajagopal et al., 2021; Das et al., 2022).

Our framework is arguably the easiest to understand, implement, and reproduce. We do not modify the original model nor require additional computation beyond fitting white boxes and additional pass over training data to extract representations (which may be done offline). We thus avoid expensive operations required by prior work, such as approximating inverted Hessian gradients (Koh and Liang, 2017) or training a network from scratch (Rajagopal et al., 2021; Das et al., 2022). Unlike prior work, the wrapper box framework is designed to be dataset, model, and task-agnostic.

Model Auditing/Algorithmic Recourse Model auditing and algorithmic recourse are commonly cited goals for fair and accountable AI systems and, thus, are closely related to explainability (Deck et al., 2024). Model auditing (Bandy, 2021; Brown et al., 2021; Yang et al., 2023) involves systematically examining a model's behavior to identify problematic behaviors, potential biases, and errors in the training data. A natural next step is algorithmic recourse (Karimi et al., 2022), which emphasizes providing actionable explanations and recommendations to users unfavorably treated by automated systems. By faithfully attributing model decisions to specific training data, wrapper boxes provide an avenue for contesting unjust decisions to support algorithmic recourse.

3 Example-based Explanation Tradeoffs

We focus on interpretable predictive models that tie inference directly to specific training examples, enabling each prediction to be faithfully explained via those same training examples that determined the model's prediction. Appendix D further discusses user perceptions of machine-retrieved examples.

To better elucidate the design space for working with such models, this section illustrates possible tradeoffs between three key variables of interest: predictive performance, explanation *faithfulness*, and explanation *simplicity*. Following Jacovi and Goldberg (2020), we conceptually define faithfulness as how accurately presented explanations reflect the actual reasoning process of the inference model. Concretely, we evaluate the faithfulness of example-based explanations by *completeness* (Gu et al., 2023), where derived examples are faithful to the extent that all instances that support the test prediction are selected. The simplicity of example-based explanations can be intuitively quantified as the number of presented instances (Nguyen and Martínez, 2020).

Section 4 discusses how such tradeoffs can be operationalized in practice for our specific wrapper box models.

3.1 Conceptual Tradeoffs

Given an input, assume the prediction model consults n training examples to make a prediction. Furthermore, assume that $m <= n$ of these training examples are shown to explain the prediction. When $m = n$, this explanation is fully faithful to the actual prediction. However, if n is very large, showing all $m = n$ of these training examples to explain the prediction may induce *cognitive overload*, often also referred to as *information overload* (Marois and Ivanoff, 2005; Abdul et al., 2020).

To simplify the explanation, one could reduce it to a smaller subset of $m < n$ of the training examples used in prediction. However, this would compromise explanation fidelity. Alternatively, the number of training examples n used in prediction could be reduced. With a smaller n, all $m = n$ examples could be shown, boosting explanation simplicity while preserving fidelity, but possibly at the cost of reduced performance.

Table 2 further illustrates the range of possible tradeoffs by presenting three scenarios, Cases I-III.

Case I attains high predictive performance and explanation fidelity, but sacrifices explanation simplicity. Here, all relevant training examples are used for both prediction and justification, thereby optimizing performance while ensuring fully faithful explanations. However, explaining model predictions via a large number of training examples However, explaining model predictions via a large number of training examples can induce information overload, hurting explanation simplicity.

Case II achieves high predictive performance and explanation simplicity but sacrifices explanation fidelity. Like Case I, all relevant training examples are used to make the prediction, maximizing performance. However, to simplify the explanation, only a subset of the training examples used to make the prediction is used to explain it. While this simplifies the explanation for the user, it sacrifices explanation fidelity to achieve this.

Case	Influential Examples	Explanation Examples	Performance	Faithfulness	Simplicity
I	All Relevant	All Relevant	✓	✓	○
II	All Relevant	Subset	✓	○	✓
III	Subset	Subset	○	✓	✓

Table 2: Case-based models permit tradeoffs between key outcome variables – predictive performance, explanation faithfulness (or fidelity), and explanation simplicity – based on which (influential) training examples are used in making a prediction vs. to explain that prediction. Note that for any given input, different models will naturally vary in which training examples are influential in performing inference for that input.

Finally, Case III sacrifices predictive performance to optimize explanation fidelity and simplicity. In this case, only a subset of relevant training examples is used to make the prediction, reducing performance. However, the same subset used to make the prediction is also used to explain it, yielding a faithful explanation. The virtue of having fewer training examples in the explanation is its simplicity, making it easier to understand.

4 Wrapper Boxes

Our wrapper box pipeline essentially "wraps" a given neural model with one or more white box classic models fitted on extracted neural representations for inference. Note that the resultant classic models do *not* attempt to approximate the underlying neural model faithfully. While both classifiers leverage learned linearly separable neural representations, the underlying decision-making process differs. Hence, derived example-based explanations faithfully explain the inference procedure of *the wrapper boxes* (interpretable classic models), *not* the original neural model.

Post hoc methods evaluate fidelity for the neural model they seek to explain (DeYoung et al., 2020; Jacovi and Goldberg, 2020) since explanations can diverge from actual model behavior. In contrast, we leverage case-based classifiers where derived example-based explanations by construction must have been consulted during inference. Loss in fidelity can only occur intentionally if fewer training examples are shown in the explanation to reduce information overload.

4.1 Learning Feature Representations

As shown in **Figure 1**, we start with a fine-tuned neural model that acts as a task-specific encoder to learn high-quality embeddings for the input text. Whereas traditional neural models often fit linear classifiers on learned representations, we extract these representations for use by various classic, white box classifiers. This substitution thus enables prediction supported by faithful, example-based explanations and is agnostic to the neural architecture, training procedure, and data used.

Our only assumption about the neural model is the ability to extract hidden states (or some form of encoded inputs). After training, another pass is made through training data to extract hidden states per token from the penultimate layer. For our sentence-level prediction tasks, we mean pool across tokens to obtain sentence-level representations. Because wrapper boxes rely on feature encodings for prediction, we store them in a format providing fast access: in-memory `Numpy` arrays.

4.2 Wrapper Box Models

We consider three case-based models in which inference is directly linked to training examples. This means that, by design, model predictions can be faithfully and intuitively attributed to specific relevant training examples.

Building on the conceptual discussion of example-based explanations in Section 3, assume the classic model consults n training examples to make a prediction for a given input and that $m <= n$ of these training examples are shown to explain the prediction. When $m < n$ (sacrificing explanation fidelity to boost explanation simplicity), a specific consideration is how each model selects which subset of m examples to show. Intuitively, the m examples should be a representative sample of the complete set of n examples to avoid introducing bias and misleading users (Lakkaraju and Bastani, 2020). Similarly, when n is reduced (to simplify explanations while preserving $m = n$ explanation fidelity), how to select the smaller subset n of training examples is also model-specific.

k Nearest Neighbors (kNN) kNN predicts the class label for each input according to the dominant class of the k most similar training examples. The nearest neighbors consulted thus constitute faithful, example-based explanations for model predictions. The simplest, unweighted kNN model performs majority voting, whereas weighted kNN weights neighbors by proximity to the input instance.

Explanations and Tradeoffs. kNN uses $n = k$ training examples to make a prediction. While we observed relatively small performance differences across the narrow range of $k = n$ values considered above, larger n generally improve predictive performance, while smaller m will simplify explanations. Because kNN inherently orders training examples by proximity to the input, training examples can be easily downsampled, either to make predictions (reduced n) or explain them ($m < n$).

However, when $m < n$ (reducing explanation fidelity to simplify the explanation), the majority label of the m nearest neighbors could differ from that of the n nearest neighbors, making the explanation inconsistent with the prediction. In this case, it may be more intuitive to explain the prediction by the m nearest neighbors whose majority label matches that of the n nearest neighbors.

Decision Trees (DTs) Decision trees learn a set of rules that act as hyperplanes. Given an input, these rules specify a decision path from the root to a given leaf node. Prediction is based on a majority vote over all training examples assigned to that leaf node. Once constructed, a DT requires the least computation for prediction since decision rules are just simple conditionals. One could even discard all training data after DT construction since only the majority label per leaf node and the final set of rules are needed for inference. However, training data must be kept if we wish to provide example-based explanations (Caruana et al., 1999).

Explanations and Tradeoffs. Just as kNN labels an input by a majority vote of the k nearest training examples, DT uses a similar vote of the given leaf node's training examples. In both cases, these training instances constitute faithful example-based explanations of the model's prediction. However, whereas kNN directly selects training examples by similarity to the input, the similarity of leaf node training examples to the input is less direct.

Because the number of training examples n used to make a prediction (for a given leaf node) may be large, faithfully showing all $m = n$ of the training examples may induce information overload. Just as kNN downsampling would intuitively select the training examples most similar to the input, DT downsampling would also select the most central training examples in the leaf node (to represent the complete leaf set best). When $m < n$ (reducing explanation fidelity to boost simplicity), just as kNN selects the m nearest neighbors whose majority label matches the predicted label, DT selects the m most central training examples whose majority label similarly matches the prediction.

L-Means We hypothesize that instances with the same class label may naturally cluster together, assuming a high-quality feature encoding of the domain (such as learned by a fine-tuned DNN).

Inference for L-means is the simplest of all wrapper boxes: given an input, we find the closest cluster centroid and assign its label to the input. This reduces the full training set to L representative cluster centroids, which act as rudimentary *prototypes* (Hase et al., 2019; Das et al., 2022). Like DT, inference only requires the majority label of relevant training examples; training data is no longer used once cluster centroids and labels are known.

Explanations and Tradeoffs. As in ProtoTex (Das et al., 2022), cluster centroids cannot be directly shown because they are latent. Instead, we must explain model predictions via the training examples that induce each centroid and whose aggregated vote assigns the centroid label.

Like other models, when the number n of voting training examples is large, showing all n examples can induce information overload. Similar to how DT downsampling selects the most central training examples in the leaf node, L-Means downsampling selects the most central training examples in the cluster. When $m < n$ (reducing explanation fidelity to boost simplicity), just as kNN selects the m nearest neighbors whose majority label matches the predicted label, L-Means selects the m most central training examples in the cluster whose majority label likewise matches the prediction.

5 Evaluation: Prediction Performance

We first compare the predictive performance of wrapper boxes vs. underlying neural models. Because neural models forecast via linear layers, we expect wrapper boxes to benefit from this learned linear separability and perform comparably. We consider two tasks and datasets:

TOXIGEN (Hartvigsen et al., 2022) consists of offensive and benign English statements generated by GPT-3 (Brown et al., 2020). We use the 9,900 human-labeled instances, ignoring other instances without gold labels. Each instance is assigned toxicity labels on a 5-point scale. We binarize labels by mapping values 1-3 to *non-toxic* and 4-5 as *toxic* for binary classification. Based on Hartvigsen et al.'s

		BART-large				DEBERTA-large				Flan-T5-large			
		Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
TOXIGEN	Original	80.85	67.69	70.07	68.86	82.77	70.47	73.94	72.16	81.38	72.43	61.97	66.79
	KNN	+0.74	+3.10	-3.52	-0.26	+0.96	**+5.02**	**-5.63**	-0.45	0.00	-0.71	+1.41	+0.50
	DT	+0.32	**+5.08**	**-9.86**	-2.96	-0.22	**+5.62**	**-11.87**	-4.07	-0.10	+0.26	-1.05	-0.51
	L-Means	-0.96	-2.01	0.00	-1.06	-0.53	+0.65	-4.58	-1.93	-0.21	-0.04	-1.06	-0.64
E-SNLI	Original	90.28	90.27	90.27	90.27	91.75	91.84	91.76	91.78	90.85	90.82	90.82	90.82
	KNN	+0.11	+0.14	+0.12	+0.13	-0.77	**-0.84**	**-0.79**	**-0.80**	-0.62	-0.61	-0.61	-0.61
	DT	**-0.92**	**-0.90**	**-0.91**	**-0.91**	+0.18	+0.11	+0.17	+0.16	+0.01	+0.01	+0.01	+0.01
	L-Means	**-2.82**	**-1.76**	**-2.75**	**-2.61**	**-0.84**	-0.45	**-0.83**	-0.77	-0.12	-0.13	-0.12	-0.13

Table 3: % change in accuracy (acc.), precision (prec.), recall (rec.), and F1 (macro-averaged) from baseline for wrapper boxes over various transformers, using only representation from the penultimate layer. Statistically significant (see Appendix B.1 for procedure) wrapper box results are bolded, with positive results in blue and negative results in red. Table 8 shows significant differences between the baseline transformers not displayed here.

90/10 train-test split, we section off a validation set, resulting in a 70/20/10 train-eval-test split. The dataset is highly skewed, with a 3:1 ratio of benign vs. toxic speech. We employ stratified sampling to maintain this ratio in each split.

E-SNLI (Camburu et al., 2018) adds crowdsourced natural language explanations for the 569,033 English premise-hypothesis pairs originally annotated in SNLI (Bowman et al., 2015). We follow the predefined training-eval-test splits. Each split contains a balanced label distribution. Appendix G.2 compares wrapper box explanations vs. those obtained via crowdsourcing.

Models We report on three language models: BART-large (Lewis et al., 2020), DEBERTA-large (He et al., 2021), and Flan-T5-large (Chung et al., 2022), based on checkpoints from Huggingface (Wolf et al., 2020). Representations are extracted from the layer immediately preceding the linear classification head for BART-large and DEBERTA-large models. For Flan-T5, representations are extracted from the layer preceding the language generation head. Implementation details for neural and white box models are discussed in Appendix A.

5.1 Results

Results are shown in **Table 3**. Our methodology for significance testing is described in Appendix B.

Wrapper boxes perform largely comparable to baseline transformers for both datasets. For TOXIGEN, across 48 results per dataset (3 transformers x 4 wrapper boxes x 4 metrics), only 6 of the 48 (12.5%) differences are statistically significant. For 3 of the 6 cases, the wrapper box performs significantly better than the baseline. For E-SNLI, while 13 of the 48 (27%) scores show statistically significant differences, whether differences are large enough to be noticeable by users is unclear (Appendix B.3). We observe no significant differences at all with Flan-T5, though note that While DEBERTA is generally the best-performing model.

Perhaps most remarkable is that the simple L-means formulation reduces the entire training set to 2-3 examples that provide the basis for all model predictions, yet still performs competitively.

Appendices F and G respectively visualize L-means clusters and provide qualitative examples.

Results for large language models (LLMs) Appendix H conducts an ablation study that evaluates the effectiveness of wrapper boxes using representations from modern LLMs. Namely, we experiment with Llama 2-7B-Instruct (Touvron et al., 2023), Llama 3-8B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct (Jiang et al., 2023), and Gemma-7B-Instruct (Mesnard et al., 2024). Results show that wrapper boxes using zero-shot LLM representations strongly outperform baseline LLM performance across both tasks.

6 Evaluation: Training Data Attribution

The ability to attribute model decisions to specific training data enables decisions to be contested on the basis of the training data responsible. To evaluate how well wrapper boxes support this use case, we adopt Yang et al. (2023)'s task formulation of finding a subset of training data S_t that, if removed, would change the model decision for a given input. We use the same two datasets but only with DEBERTA representations (best performing model).

Baselines Yang et al. (2023)'s two algorithms are limited to convex linear classifiers (e.g., logistic regression). We report these as baselines. Appendix C.5 details our reproduction of their reported results, further validating the new results we

		TOXIGEN			E-SNLI		
Classifier	**Selector**	**↑Coverage%**	**↑Correctness%**	**↓Median**	**↑Coverage%**	**↑Correctness%**	**↓Median**
LR	Yang Fast	27.45	27.13	51.00	89.83	0.39	76,446.50
LR	Yang Slow	27.45	26.49	33.00	89.83	0.11	2.00
DT	Greedy	12.02	12.02	24.00	3.23	3.23	89.00
L-Means	Greedy	100.00	100.00	6,377.00	100.00	100.00	140,523.00
KNN	Greedy	100.00	100.00	211.00	100.00	100.00	77.50

Table 4: Benchmarking selectors to derive S_t. Coverage is the % of test inputs for which a S_t was proposed. Correctness is the % of test inputs for which a S_t was proposed and verified that their removal and retraining led to a prediction flip. Median is the median set cardinality across only the verified subsets that lead to prediction flip.

report with their methods on our own datasets.

Yang Fast (Algorithm 1) uses influence functions to estimate expected change in output probability from removing subset S_t. A S_t is only output if the expected change exceeds a threshold τ.

Yang Slow (Algorithm 2) starts with all training data and seeks to iteratively reduce size of S_t by approximating expected changes to model parameters θ upon removal. Like *Yang Fast*, S_t is only found if the expected output change exceeds τ.

Of note, Yang et al. report on five binary datasets in their work: three balanced, and two highly skewed 9:1 ("hate" and "essays"). While results are strong on the balanced datasets, coverage is low on hate (67%) and very low on essays (11-12%). Yang et al. remark upon hate's severe label skew, and to address it, select a post hoc $\tau = 0.25$ for this dataset only (using $\tau = 0.5$ for all others). Oddly, they do not note or address the same skew in essays, which may lead the very low coverage reported.

Our Approach Algorithm 1 defines a greedy approach to derive S_t from wrapper box explanations. For kNN, C^{tr} includes all neighbors of the input, ranked by proximity. For DT, C^{tr} comprises all examples in the same leaf, ranked by proximity. For L-means, C^{tr} consists of all points in the same cluster, ranked by proximity to the cluster centroid. Post-filtering, we remove examples in chunks until a prediction flip is observed. S_t is then refined (iteratively or in chunks, depending on ϕ) until no size reduction is possible. This encourages the derived S_t to be minimal (but still leads to a prediction flip). See Appendices C.1 and C.2 for further details and an optimized algorithm for kNN (no training).

Algorithm 1 Greedy approach to derive S_t from wrapper box explanations

Input: f: Model, C^{tr}: Ranked set of candidate training examples to select from, x_t: Test input, y_t: Test input label, B: Number of bins, ϕ: Iterative threshold

Output: S_t, a subset of training points that flips y_t (or $\emptyset$ if unsuccessful)

function FINDSUBSET($C^{\text{tr}}, x_t, y_t, B$)
  $b \leftarrow \lceil \frac{|\mathcal{L}|}{B} \rceil$ ▷ Bin size
  $\mathcal{L} \leftarrow \{(x_i, y_i) \in \mathcal{C}^{\text{tr}} \mid y_i = y_t\}$ ▷ Filter candidates to match prediction to reduce search complexity
  for $i \leftarrow 1$ to B **do**
    $C_i^{\text{tr}} \leftarrow C^{\text{tr}} \setminus \{\mathcal{L}[j] \mid j \leq i * b\}$
    $\hat{f} \leftarrow$ train_model(C_i^{tr})
    $\hat{y}_t \leftarrow \hat{f}(x_t)$
    if $\hat{y}_t \neq y_t$ **then**
      return $\{\mathcal{L}_j \mid j \leq i\}$
  return $\emptyset$
$S_t \leftarrow$ FINDSUBSET($C^{\text{tr}}, x_t, y_t, B$)
previous_size $\leftarrow 0$
while $|S_t| > 0$ **and** $|S_t| \neq$ previous_size **do**
  previous_size $\leftarrow |S_t|$
  if $|S_t| < \phi$ **then**
    $S_t \leftarrow$ FINDSUBSET($S_t, x_t, y_t, |S_t|$)
  else
    $S_t \leftarrow$ FINDSUBSET(S_t, x_t, y_t, B)
return S_t

6.1 Results

Results in **Table 4** report three key metrics: *coverage* (% of test inputs for which a subset S_t was proposed), *correctness* (% of test inputs for which removing S_t correctly changed the model decision), and the median size of correct S_t subsets found).

Baselines. Yang et al.'s methods do not perform well. For TOXIGEN, we suspect the issue is label skew (see discussion above). Classifying directly via DEBERTA vs. using logistic regression ($\tau = 0.5$) with DEBERTA representations yielded comparable results (Table 6), so we use $\tau = 0.5$ for Yang et al. (2023)'s methods on TOXIGEN.

For E-SNLI, Yang Fast/Slow propose $S_t \sim 90\%$ of the time, but removing S_t almost never changes model decisions. Because they only consider binary classification tasks, their formulation with τ likely does not make sense for multi-class tasks like E-SNLI that typically involve predicting the most probable class through softmax probabilities.

Wrapper boxes. Overall, kNN is the clear winner, with perfect coverage and correctness and far smaller S_t than L-means. While both kNN and L-means achieve perfect coverage and correctness

on both datasets, S_t tends to be quite large for L-means since clusters (see Appendix F) are mostly homogeneous; many training supporting the model decision must be removed before points with other labels come to the fore to change the decision.

DT has low coverage because its subset candidate search space is so small, having only leaf examples. This contrasts sharply with kNN (all training examples) and L-means (all cluster points). By the same token, when DT does find a S_t subset, it tends to be far smaller than kNN or L-means.

7 Conclusion

We propose wrapper boxes to provides faithful, example-based explanations for classic case-based model predictions, attributing decisions to specific training data. Our proposed pipeline is quite general and agnostic to the underlying neural architecture, training procedure, and input data. After training a neural model, the learned feature representation is input to white-box case-based reasoning models for prediction. Because case-based models tie inference directly to specific training data, each prediction can be faithfully attributed to the training examples responsible.

Our first evaluation showed that white case-based models could deliver predictive performance largely comparable to baseline transformers, as seen across seven large pre-trained language models, two datasets of varying scale, three classic models, and four metrics.

In addition, we discussed how such attribution enables automated decisions to be contested based on the training data responsible for those decisions. In comparison to prior work (Yang et al., 2023), our approach achieves both higher coverage and correctness in identifying which training data to remove to change a model decision.

Beyond contesting model decisions, other use cases include intuitively explaining decisions to end-users based on past examples or supporting data-centric AI operations for model developers (e.g., training data augmentation and cleaning).

8 Limitations

8.1 Time and Space Requirements

Wrapper boxes require additional space to store training instances to be presented as example-based explanations. For example, while DT and L-means models no longer require training data for inference once trained, they must continue to store training data to provide example-based explanations. For DT, representative subsets of examples per leaf node may be pre-computed and cached ahead of time for fast explanation retrievals. L-means is similar: since clusters are invariant across all predictions, representative subsets of desired sizes may be pre-computed and cached ahead of time for fast explanation retrievals at inference time. In both cases, storage demands vary depending on the number of desired examples to present for explanations.

Different wrapper boxes will naturally vary in computation time and space needs, with some models potentially resulting in slower or faster inference than the base neural model. Moreover, we have used relatively simple implementations for each wrapper box. More advanced schemes, e.g., dynamic k for kNN (Zhang et al., 2018), could further increase the computational time or space requirements. Generally, standard computational requirements of classic models are carried forward into our adoption of them as wrapper boxes.

8.2 Use-Cases for Training Data Attribution

The ability of wrapper boxes to faithfully attribute model decisions to specific training data has a variety of applications. However, our study only evaluates how well wrapper boxes enable model decisions to be contested based on the training data responsible for those decisions. More specifically, we considered the task of identifying which training data would need to be removed in order to change a model decision (Yang et al., 2023) (Section 6).

Beyond contesting model decisions, other use-cases include explaining decisions to end-users based on known past examples (Schank et al., 2014). Attribution could also support data-centric AI operations for model developers to help uncover artifacts (Lertvittayakumjorn and Toni, 2021), errors (Koh and Liang, 2017), and gaps (Khanna et al., 2019) in training data, addressed by label cleaning (Teso et al., 2021; Northcutt et al., 2021) data augmentation (Feng et al., 2021), and other *data-centric* operations (Anik and Bunt, 2021).

Section 2 noted that prior work has consistently validated the significance, utility, and effectiveness of example-based explanations for users (Aamodt and Plaza, 1994; Sørmo et al., 2005; Richter and Weber, 2016). Benefits include increased model understanding (simulatability), complementary performance, and trust. (Yeh et al., 2018; Papernot and McDaniel, 2018; Cai et al., 2019; Hase and Bansal, 2020; Han et al., 2020; Rajagopal et al., 2021; Das

et al., 2022; Suresh et al., 2022; Chen et al., 2023). However, we have yet to actually evaluate any of these benefits in the context of wrapper boxes.

For both use cases above – explaining model decisions to end-users or supporting data-centric AI for model developers – user studies would be valuable to assess the utility of wrapper boxes.

Of particular interest, Section 3 discussed how wrapper boxes permit thoughtful tradeoffs across three key variables of interest: predictive performance, explanation *faithfulness*, and explanation *simplicity*. However, we have yet to investigate these tradeoff space with real users. Future work could conduct user studies to better elucidate how different tradeoffs impact real user experience.

8.3 S_t Assumptions and Challenges

While Section 6 usefully investigates how model decisions can change by counterfactually removing a subset of training data S_t, many more counterfactual conditions could be considered that would also alter model decisions, such as the choice of model and training regieme. Neither we nor Yang et al. (2023) consider such counterfactual conditions that would be more difficult for end-users to contest.

Similarly, we and Yang et al. both apply a classification model atop fixed feature representations, without considering counterfactual data conditions that would change feature representations. In Yang et al.'s work, bag-of-words and BERT embeddings are used off-the-shelf and counterfactual training data conditions only impact the learned logistic regression model. In our work, neural representations are fine-tuned on all training data and counterfactual training data conditions only impact wrapper box inference atop neural representations.

As S_t grows, model auditing becomes more difficult, akin to the cognitive overload of showing many examples in a model explanation (Section 3.1). However, prior work (Ilyas et al., 2022; Yang et al., 2023) has shown that large S_t can also indicate predictor robustness, since more training data must be removed to change model decisions. Future work could thus usefully explore tradeoffs of benefits between small vs. large S_t subsets.

Acknowledgments

This research was supported in part by Cisco and by *Good Systems* [1], a UT Austin Grand Challenge to develop responsible AI technologies. This work was also partially supported by NSF grant IIS-2107524. The statements made herein are solely the opinions of the authors and do not reflect the views of the sponsoring agencies.

[1] http://goodsystems.utexas.edu/

References

Agnar Aamodt and Enric Plaza. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

Ashraf Abdul, Christian von der Weth, Mohan Kankanhalli, and Brian Y Lim. 2020. COGAM: Measuring and moderating cognitive load in machine learning model explanations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–14, New York, NY, USA. Association for Computing Machinery.

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

M A Ahmad, C Eckert, and A Teredesai. 2018. Interpretable machine learning in healthcare. *Proceedings of the 2018 ACM*.

Ariful Islam Anik and Andrea Bunt. 2021. Data-centric explanations: Explaining training data of machine learning systems to promote transparency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, number Article 75 in CHI '21, pages 1–13, New York, NY, USA. Association for Computing Machinery.

Jackie Ayoub, X. Jessie Yang, and Feng Zhou. 2021. Combat covid-19 infodemic using explainable natural language processing models. *Information Processing & Management*, 58(4):102569.

Jack Bandy. 2021. Problematic machine behavior: A systematic literature review of algorithm audits. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1):1–34.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115.

S Basu, P Pope, and S Feizi. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*.

Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

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, pages 648–657, New York, NY, USA. Association for Computing Machinery.

Reuben Binns, Max Van Kleek, Michael Veale, Ulrik Lyngs, Jun Zhao, and Nigel Shadbolt. 2018. 'it's reducing a human being to a percentage': Perceptions of justice in algorithmic decisions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–14, New York, NY, USA. Association for Computing Machinery.

Samuel R. 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, Lisbon, Portugal. Association for Computational Linguistics.

Shea Brown, Jovana Davidovic, and Ali Hasan. 2021. The algorithm audit: Scoring the algorithms that score us. *Big Data & Society*, 8(1):2053951720983865.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Carrie J Cai, Jonas Jongejan, and Jess Holbrook. 2019. The effects of example-based explanations in a machine learning interface. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, pages 258–262, New York, NY, USA. Association for Computing Machinery.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

R Caruana, H Kangarloo, J D Dionisio, U Sinha, and D Johnson. 1999. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, pages 212–215, United States.

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

J. T. Casagrande, M. C. Pike, and P. G. Smith. 1978. An improved approximate formula for calculating sample sizes for comparing two binomial distributions. *Biometrics*, 34(3):483–486.

Valerie Chen, Q Vera Liao, Jennifer Wortman Vaughan, and Gagan Bansal. 2023. Understanding the role of human intuition on reliance in human-AI decision-making with explanations. *Proc. ACM Hum.-Comput. Interact.*, 7(CSCW2):1–32.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Anubrata Das, Chitrank Gupta, Venelin Kovatchev, Matthew Lease, and Junyi Jessy Li. 2022. ProtoTEx: Explaining model decisions with prototype tensors. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2986–2997, Dublin, Ireland. Association for Computational Linguistics.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate speech dataset from a white supremacy forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20, Brussels, Belgium. Association for Computational Linguistics.

Luca Deck, Jakob Schoeffer, Maria De-Arteaga, and Niklas Kühl. 2024. A critical survey on fairness benefits of explainable AI. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pages 1579–1595, 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.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Jonathan Dodge. 2022. Position: The case against case-based explanation. In *IUI Workshops*, pages 175–180.

Jonathan Dodge, Q. Vera Liao, Yunfeng Zhang, Rachel K. E. Bellamy, and Casey Dugan. 2019. Explaining models: An empirical study of how explanations impact fairness judgment. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, page 275–285, New York, NY, USA. Association for Computing Machinery.

Mengnan Du, Ninghao Liu, and Xia Hu. 2019. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.

Alec Go. 2009. Twitter sentiment classification using distant supervision. *CS224N project*.

Peijian Gu, Yaozong Shen, Lijie Wang, Quan Wang, Hua Wu, and Zhendong Mao. 2023. IAEval: A comprehensive evaluation of instance attribution on natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11966–11977, Singapore. Association for Computational Linguistics.

Ben Hamner, Jaison Morgan, lynnvandev, Mark Shermis, and Tom Vander Ark. 2012. The hewlett foundation: Automated essay scoring.

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, Online. Association for Computational Linguistics.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. 2019. Interpretable image recognition with hierarchical prototypes. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 7(1):32–40.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9525–9587. PMLR.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Ziwei Ji, Justin Li, and Matus Telgarsky. 2021. Early-stopped neural networks are consistent. In *Advances in Neural Information Processing Systems*, volume 34, pages 1805–1817. Curran Associates, Inc.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Margot E Kaminski. 2019. The right to explanation, explained. *Berkeley Technol. Law J.*, 34(1):189–218.

Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2022. A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Comput. Surv.*, 55(5):1–29.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.*, 30.

Mark T. Keane and Eoin M. Kenny. 2019. How case-based reasoning explains neural networks: A theoretical analysis of xai using post-hoc explanation-by-example from a survey of ann-cbr twin-systems. In *Case-Based Reasoning Research and Development*, pages 155–171, Cham. Springer International Publishing.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. 2019. Interpreting black box predictions using fisher kernels. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 3382–3390. PMLR.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Janet Kolodner. 2014. *Case-Based Reasoning*. Morgan Kaufmann.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Himabindu Lakkaraju and Osbert Bastani. 2020. "how do i fool you?": Manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 79–85, New York, NY, USA. Association for Computing Machinery.

Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. 2021. What do we want from explainable artificial intelligence (XAI)? – a stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artif. Intell.*, 296:103473.

Piyawat Lertvittayakumjorn and Francesca Toni. 2021. Explanation-based human debugging of NLP models: A survey. *Transactions of the Association for Computational Linguistics*, 9:1508–1528.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Han Liu, Yizhou Tian, Chacha Chen, Shi Feng, Yuxin Chen, and Chenhao Tan. 2023. Learning human-compatible representations for case-based decision support. *arXiv preprint arXiv:2303.04809*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc interpretability for neural nlp: A survey. *ACM Comput. Surv.*, 55(8).

René Marois and Jason Ivanoff. 2005. Capacity limits of information processing in the brain. *Trends Cogn. Sci.*, 9(6):296–305.

Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342.

Thomas Mesnard et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Edoardo Mosca, Ferenc Szigeti, Stella Tragianni, Daniel Gallagher, and Georg Groh. 2022. SHAP-based explanation methods: A review for NLP interpretability. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4593–4603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Lukas Muttenthaler, Jonas Dippel, Lorenz Linhardt, Robert A. Vandermeulen, and Simon Kornblith. 2023. Human alignment of neural network representations. *arXiv preprint arXiv:2211.01201*.

Allan Newel and Herbert A Simon. 1972. Human problem solving. *Englewood Cliffs, NJ*.

An-Phi Nguyen and María Rodríguez Martínez. 2020. On quantitative aspects of model interpretability. *arXiv [cs.LG]*.

Ian E. Nielsen, Dimah Dera, Ghulam Rasool, Ravi P. Ramachandran, and Nidhal Carla Bouaynaya. 2022. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Processing Magazine*, 39(4):73–84.

Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Nicolas Papernot and Patrick McDaniel. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930. Curran Associates, Inc.

Dheeraj Rajagopal, Vidhisha Balachandran, Eduard H Hovy, and Yulia Tsvetkov. 2021. SELFEXPLAIN: A self-explaining architecture for neural text classifiers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 836–850, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, 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.

Michael M Richter and Rosina O Weber. 2016. *Case-based reasoning*. Springer.

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

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16(none):1 – 85.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Roger C Schank, Alex Kass, and Christopher K Riesbeck. 2014. *Inside case-based explanation*. Psychology Press.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Donghee Shin. 2021. The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI. *Int. J. Hum. Comput. Stud.*, 146:102551.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Karen Spärck Jones. 1974. Automatic indexing. *Journal of documentation*, 30(4):393–432.

Agus Sudjianto and Aijun Zhang. 2021. Designing inherently interpretable machine learning models. *arXiv preprint arXiv:2111.01743*.

Xiaobing Sun and Wei Lu. 2020. Understanding attention for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3418–3428, Online. Association for Computational Linguistics.

Harini Suresh, Kathleen M Lewis, John Guttag, and Arvind Satyanarayan. 2022. Intuitively assessing ml model reliability through example-based explanations and editing model inputs. In *27th International Conference on Intelligent User Interfaces*, IUI '22, page 767–781, New York, NY, USA. Association for Computing Machinery.

Frode Sørmo, Jörg Cassens, and Agnar Aamodt. 2005. Explanation in case-based Reasoning–Perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143.

Stefano Teso, Andrea Bontempelli, Fausto Giunchiglia, and Andrea Passerini. 2021. Interactive label cleaning with example-based explanations. In *Advances in Neural Information Processing Systems*, volume 34, pages 12966–12977. Curran Associates, Inc.

Mariya Toneva and Leila Wehbe. 2019. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Eric Wallace, Shi Feng, and Jordan Boyd-Graber. 2018. Interpreting neural networks with nearest neighbors.

In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 136–144, Brussels, Belgium. Association for Computational Linguistics.

Eric Wallace, Matt Gardner, and Sameer Singh. 2020. Interpreting predictions of NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 20–23, Online. Association for Computational Linguistics.

Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1543–1552, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Xinru Wang and Ming Yin. 2021. Are explanations helpful? a comparative study of the effects of explanations in AI-assisted decision-making. In *26th International Conference on Intelligent User Interfaces*, IUI '21, pages 318–328, New York, NY, USA. Association for Computing Machinery.

Zhiguo Wang, Wael Hamza, and Linfeng Song. 2017. k-nearest neighbor augmented neural networks for text classification. *CoRR*, abs/1708.07863.

Mingtong Wen, Tingyu Xia, Bowen Liao, and Yuan Tian. 2023. Few-shot relation classification using clustering-based prototype modification. *Knowledge-Based Systems*, 268:110477.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Jinghan Yang, Sarthak Jain, and Byron C Wallace. 2023. How many and which training points would need to be removed to flip this prediction? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2571–2584, Dubrovnik, Croatia. Association for Computational Linguistics.

Jinghan Yang, Linjie Xu, and Lequan Yu. 2024. Relabeling minimal training subset to flip a prediction. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1085–1098, St. Julian's, Malta. Association for Computational Linguistics.

Chih-Kuan Yeh, Joon Sik Kim, Ian E H Yen, and Pradeep Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9311–9321, Red Hook, NY, USA. Curran Associates Inc.

Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. 2018. Efficient knn classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1774–1785.

Wei Zhang, Ziming Huang, Yada Zhu, Guangnan Ye, Xiaodong Cui, and Fan Zhang. 2021. On sample based explanation methods for NLP: Faithfulness, efficiency and semantic evaluation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5399–5411, Online. Association for Computational Linguistics.

Jianlong Zhou, Amir H. Gandomi, Fang Chen, and Andreas Holzinger. 2021. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5).

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, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dataset	Train	Valid	Test	Ratio
TOXIGEN	6980	1980	940	3:1
E-SNLI	549361	9842	9824	balanced

Table 5: Dataset information. Ratio is the distribution of labels in each split (same for all splits due to stratified sampling).

A Implementation Details

All transformers are fine-tuned using the AdamW optimizer (Loshchilov and Hutter, 2017) on Cross Entropy loss over ten epochs, with early stopping (Ji et al., 2021) if validation performance degrades for two consecutive evaluations (every 100 steps). All layers are fine-tuned. We use seed 42, a learning rate of $1e-5$, and a batch size of 16 for BART and DEBERTA. Since Flan-T5 is larger, we use a learning rate of $1e-4$ and a batch size of 8. No hyperparameter search is conducted. All models are fine-tuned on a single compute node with three NVIDIA A100 GPUs and 256GB of DDR4 RAM within one week of GPU hours.

Feature encodings are extracted from each trained neural model for use by classic models (i.e., wrapper boxes). We implement Logistic Regression (LR), KNN, and L-means using Scikit-Learn (Pedregosa et al., 2011). Early experiments suggest that results were fairly comparable across small values of k (1, 3, 5, 7, and 9) for unweighted and weighted. For this reason, we report unweighted KNN with `k=5`. We utilize K-D trees (Bentley, 1975) for efficiently retrieving nearest neighbors. For decision trees, we use `DecisionTreeClassifier` from Scikit-Learn and set `max_depth` = 3 to guard against over-fitting for skewed TOXIGEN, though this value was not tuned. For E-SNLI, since the Scikit-Learn implementation does not scale well, we opt for LightGBM (Ke et al., 2017) with one classifier. For both trees, we set the minimum number of samples in each leaf to be 20. For L-means, we set `algorithm='elkan'` for more efficient computation since our clusters are well-defined. We use $\tau = 0.5$ for LR on TOXIGEN and set `multi_class=multinomial` for E-SNLI. Similarity between data instances in the feature space is measured via Euclidean distance. All results are single-run with random seed 42.

The number of training points and the distribution of labels in each split is shown in Table 5. TOXIGEN[2] is skewed with a 3:1 skew for benign vs. toxic examples, respectively. E-SNLI[3] is balanced. For E-SNLI, we excluded 6 training pairs containing blank hypotheses.

B Significance Testing

B.1 Procedure

We perform statistical testing to A/B test the performance of baseline transformers vs. treatment wrapper boxes. We also apply the same procedure to compare baseline transformers to each other. Specifically, correct vs. incorrect predictions by each model yield separate binomial distributions. Given relatively large sample sizes, we compute the z-score as shown below (Casagrande et al., 1978):

$$z = \frac{\hat{m}_1 - \hat{m}_2}{\sqrt{\hat{m} - (1-\hat{m})(\frac{1}{n_1} + \frac{1}{n_2})}} \quad (1)$$

where $\hat{m} = \frac{n_1\hat{m}_1 + n_2\hat{m}_2}{n_1+n_2}$. We test the null hypothesis that for some given metric m (e.g. accuracy), there is no significant difference between the two binomial distributions, baseline vs. treatment, or $m_1 = m_2$ (alternatively $m_1 \neq m_2$). We use $\alpha = 0.05$ where results with $p < \alpha$ are significant. We bold and color code significantly different results in Table 3. Each cell denotes a comparison between a white wrapper box (row) with respect to a baseline transformer (column category) for a particular metric (column type). For transparency, Appendix B.2 shows all significance test p-values.

B.2 Ablation Results

Table 8 displays the p-values for all pairwise comparisons between baseline transformers across four metrics (accuracy, precision, recall, F1) on two datasets (TOXIGEN, E-SNLI). **Table 9** shows the p-values for all comparisons between wrapper boxes (rows) and their corresponding baseline transformers (column categories) for the four metrics (columns) on two datasets (row categories).

B.3 Are Significant Differences Noticeable?

In regard to the experimental practice of significance testing, we also wanted to raise a more subtle point here. In system-centered evaluations, we are accustomed to A/B testing of baseline vs. treatment conditions and measuring the statistical significance of observed differences. Indeed, we followed this experimental paradigm in this work,

[2] https://huggingface.co/toxigen

[3] https://huggingface.co/esnli

showing that wrapper boxes perform largely comparable to that of the underlying neural models.

This experimental paradigm is well-motivated from the standpoint of continual progress, that small but significant differences from individual studies will add up over time to more substantial gains. However, with regard to an individual study, small, statistically significant differences are typically unobservable to users in practice, especially when a slightly less performant system offers some other notable capability, such as providing explanations as well as predictions.

As a result, an interpretable system that is less performant according to system-based performance metrics may still be experienced as equally performant. Spärck Jones (1974) famously remarked that "statistically significant performance differences may be too small to be of much operational interest", proposing her classic rule of thumb that only improvements of 5% or more are *noticeable*, while improvements of 10% or more are *material*.

The upshot is that while we are accustomed to placing great weight on minute but statistically significant differences between conditions in system-centered evaluations, from the standpoint of user experience, we should be mindful that such small differences will often be invisible to users, who may well prefer an interpretable system that seems equally performant, even if it actually performs worse by statistical significance testing.

Dataset	Classifier	Acc.	Prec.	Rec.	F1
TOXIGEN	Original	82.77	70.47	73.94	72.16
	LR	-0.22	+4.12	**-9.86**	-3.23
E-SNLI	Original	91.75	91.84	91.76	91.78
	LR	+0.38	+0.29	+0.36	+0.35

Table 6: % change in accuracy, precision, recall, and F1 (macro-averaged) for logistic regression (LR) using DEBERTA penultimate representations on TOXIGEN and E-SNLI. Like wrapper boxes, logistic regression with DEBERTA penultimate representations also performs comparably to the original, underlying neural model.

C Training Data Attribution Clarifications and Results

C.1 Iterative vs Chunked Removal

Refitting a new classifier can be expensive, especially for the larger E-SNLI dataset, when repeated many times. For example, if we were to iteratively remove ranked cluster examples for L-means on E-SNLI, and it takes (empirically) approximately 15 seconds to retrain and obtain a new test prediction, then finding S_t would take approximately a month on a single node. Hence, in practice, we chunk ranked examples into B consecutive bins such that removal occurs simultaneously for all points in the same bin. Once a S_t is identified this way, we recursively refine iteratively when the subset is less than some iterative threshold ϕ, or further split the candidate S_t into smaller B bins in a chunked fashion. Of course, there is likely an efficiency-performance tradeoff here associated with the numbers of bins and ϕ. As the number of bins increases (smaller chunks), subsets should be minimal but demands more computation. Vice versa, a subset may always be identified (e.g. if the number of bins equals 1, where we are removing the entire candidate set of training points), but it may not be very useful for model auditing. In Section 6, on both datasets, for DT and L-means, we employ 10 bins (each bin thus consists of 10% of the training data) and set the iterative threshold to be $\phi = 100$ examples (only do chunk removal when candidate S_t is above this threshold).

To give some qualitative runtimes (on a single node with a 2.1GHz, 48-core Intel Xeon Platinum 8160 "Skylake" CPU) to highlight the infeasibility of iterative removing and retraining for E-SNLI, Yang fast takes 3 minutes per example, Yang slow 15 minutes per example, DT 5 minutes per example, and L-means 25 minutes per example. kNN is the fastest and finds S_t per example in under 1 second since it requires no retraining (see Algorithm 2 detailed in Appendix C.2 below).

C.2 Finding Subsets for KNN

kNN is a special white box classifier in that there is no "training". The inference module simply remembers the training examples and their labels, while computing nearest neighbors on-demand, given test inputs. When deriving S_t, we were thus able to 1) precompute and cache all neighbors and 2) perform iterative "removal" to assess prediction flip without retraining. Specifically, given a test input, we first obtain the ranked list of all neighbors (training points) by proximity. Like algorithm 1, neighbors are then filtered down to only those with the same label as the prediction as candidates to remove. After filtering, we iteratively remove the nearest neighbor, and then directly examine the next k nearest neighbors to obtain the new prediction. We can do this because our implementation of kNN is unweighted, where it makes predictions

Algorithm 2 Optimized greedy approach to derive S_t for kNN

Input: f: Model, C^{tr}: Ranked set of candidate training examples to select from, x_t: Test input, y_t: Test input label

Output: S_t, a subset of training points that flips y_t (or $\emptyset$ if unsuccessful)

1: $\mathcal{L} \leftarrow \{(x_i, y_i) \in \mathcal{C}^{\text{tr}} \mid y_i = y_t\}$ ▷ Filter candidates to match prediction to reduce search complexity
2: **for** $i \leftarrow 1$ to $|\mathcal{L}|$ **do**
3: $\quad C_i^{\text{tr}} \leftarrow C^{\text{tr}} \setminus \{\mathcal{L}[j] \mid j \leq i\}$
4: $\quad \hat{y}_t \leftarrow$ majority_vote(C_i^{tr}, k) ▷ Predict using the k nearest neighbors
5: $\quad$ **if** $\hat{y}_t \neq y_t$ **then**
6: $\quad\quad$ **return** $\{\mathcal{L}_j \mid j \leq i\}$
7: **return** $\emptyset$

by majority vote using the k nearest neighbors. We can thus easily observe if a prediction flip occurred by monitoring if the majority training label in a k-sized window has changed without re-training. This makes the greedy approach to identify S_t for kNN the fastest selector amongst all others considered in Section 6.

C.3 Extending Our Greedy Selection Algorithm to Influence Functions?

Algorithm 1 is agnostic to the inference model and the ranking procedure, as long as both are available. From this perspective, one can theoretically leverage influence functions (IF) (Koh and Liang, 2017) to obtain training examples ranked by influence to run the procedure. However, because IF assumes linearity and twice-differentiable loss functions, this constrains their application to non-convex white wrapper boxes (e.g., kNN). This restriction is why Yang et al. limit their analysis to logistic regression alone. One could apply IF to the underlying neural module, but it is computationally infeasible to repeatedly retrain the language models analyzed in this study. Our Algorithm 1 is feasible in our experiments because our wrapper boxes are lightweight, e.g., kNN does not require retraining!

C.4 Comparisons to Prior Work

Algorithm 1 is similar to *data models* (Ilyas et al., 2022) in that some model retraining is required. However, we note several distinctions here. First, data models are *surrogate models* trained to approximate the output of a black-box neural model. Thus, subsets identified through data models are not guaranteed to lead to a prediction flip, whereas we are guaranteed that resultant subsets are correct. Second, learning data models requires collecting labels (probability outputs) from the neural module retrained with different subsets of the training set. This is considerably more expensive since we only retrain lightweight white box models. Third, data models are affected by the stochastic nature of the neural model and its supervised learning framework, so its outputs are nondeterministic and only approximate the neural. On the other hand, our approach yields deterministic subsets, since the re-trained wrapper model must have the same hyperparameters as the original.

Overall, this paper and prior work (Ilyas et al., 2022; Yang et al., 2023) have shown that finding S_t is computationally challenging. Our greedy approach (besides kNN) requires retraining a new white box classifier for each removal, Yang et al. (2023) necessitates inverse hessian approximations, and Ilyas et al. (2022) requires numerous retraining of models to obtain labels for data models. Despite these computational demands, none of these approaches guarantee that identified subsets are minimal (smallest possible) or unique (for each test input). An alternative direction, as investigated in (Yang et al., 2024), is to flip training labels instead of removing whole examples. While this method is sample-efficient for binary classification tasks, its efficacy in multiclass tasks remains uncertain.

C.5 Reproducing Yang et al. (2023)

Our results in Section 6 show that both selectors from Yang et al. (2023) perform poorly on our two selected datasets. To demonstrate that the baseline was implemented correctly (and thereby validate our baseline results with Yang et al.'s methods in our own experiments), we reproduce reported results on the datasets evaluated in Yang et al. (2023), including Movie sentiment[4] (Socher et al., 2013); Twitter sentiment classification[5] (Go, 2009); Essay grading[6] (Hamner et al., 2012); Emotion classification[7] (Saravia et al., 2018), and; Hate speech detection[8] (de Gibert et al., 2018). We adopt their source code[9] to reproduce their results.

[4] https://github.com/successar/instance_attributions_NLP/tree/master/Datasets/SST
[5] https://www.kaggle.com/datasets/kazanova/sentiment140
[6] https://www.kaggle.com/competitions/asap-aes/data
[7] https://huggingface.co/datasets/dair-ai/emotion
[8] https://huggingface.co/datasets/odegiber/hate_speech18
[9] https://github.com/ecielyang/Smallest_set

Dataset	Selector	Coverage	Correctness	Median
Movie reviews	Yang Fast	64.22	64.11	94
	Yang Slow	64.22	63.19	76
Essays	Yang Fast	7.47	7.47	24
	Yang Slow	7.47	7.16	12
Emotion	Yang Fast	71.78	71.78	64
	Yang Slow	71.78	70.79	51
Hate speech	Yang Fast	52.94	52.66	135
	Yang Slow	46.41	44.16	103
Tweet Sentiment	Yang Fast	89.80	89.50	110
	Yang Slow	75.30	60.30	213

Table 7: Coverage, Correctness, and Median for logistic regression using BERT `[cls]` representations for the two S_t selector algorithms proposed in Yang et al. (2023). Note that results differ slightly from their Table 2 due to stochastic differences in generating dataset splits and potential differences in L2 penalty term.

As part of reproduction work, we share a few data-cleaning details not reported in Yang et al. (2023). Although their training and testing split sizes are provided in their Table A1, the random seed used to generate those splits was unavailable (besides movie reviews, which appear to use the provided train split and the validation split for testing). Consequently, we observe slightly different subset results than those reported in their Table 2.

To preserve the split sizes reported in their Table A1, we use the the movie review dataset training split as-is while using the provided validation set as-is for testing. We only use the provided training set for all other datasets and break it down to a 90/10 train-test split. For essays, we first binarize the training split dataset by converting the top 10% of essay scores to 1 and the rest to 0. Only examples labeled with "sadness" (0) and "joy" (1) were kept for emotions. For hate speech, we similarly only kept training examples labeled with "nohate" (0) and "hate" (1). 19,000 random training points were sampled from the tweet sentiment training split.

Representations using the `[cls]` token of `bert_base_uncased`[10] (Devlin et al., 2019) were extracted for each dataset, following (Yang et al., 2023). For recency, we did not reproduce results with bag-of-words embeddings. We then fitted a logistic regression for each dataset using the extracted BERT representations, using $\tau = 0.25$ for hate speech and $\tau = 0.5$ for all other datasets as specified in (Yang et al., 2023).

Table 7 shows the subset results for various classifiers and selector methods on the five datasets. We apply the same metrics as described in Table 4, noting that Coverage and Correctness are equivalent to the columns "Found S_t" and "Flip Successful" in Yang et al. (2023)'s Table 2.

Our reproduced results are comparable to their reported results, barring stochastic differences due to different train/test splits and potentially different L2 penalty terms for each fitted regressor. Furthermore, as remarked in their limitations, "assuming a stochastic parameter estimation method (e.g., SGD) the composition of S_t may depend on the arbitrary random seed, similarly complicating the interpretation of such sets," so it is not surprising that we observe somewhat different outcomes.

D AI vs. Human Perceptions of Similarity

In explaining model decisions to end-users by attributing decisions to specific training data, we assume that users will understand why the given training examples shown are relevant to a given input. Otherwise, the training examples shown could appear spurious, and users might not understand why the model deemed these training examples relevant to the input at hand. This raises two key questions: 1) how do wrapper boxes measure instance similarity, and 2) how closely does this measurement align with human perceptions of instance similarity?

Given the neural model's extracted embeddings, wrapper boxes compute similarity between instances via Euclidean distance. Instance similarity is thus assessed as a combination of the embedding space and the distance function.

[10] https://huggingface.co/google-bert/bert-base-uncased

Human perceptual judgments may naturally diverge from large pre-trained language models' learned latent representation space. For example, Liu et al. (2023) show that nearest neighbor images using ResNet (He et al., 2016) representations may not align with human similarity judgments. However, they also show that more human-aligned representations can be learned to improve human decision-making. Similarly, Toneva and Wehbe (2019) show that modifying BERT (Devlin et al., 2019) to match human brain recordings better enhances model performance.

We hypothesize that continuing progress in developing increasingly powerful pre-trained large language models will naturally trend toward producing representations having greater alignment with human perceptions (Muttenthaler et al., 2023). Of course, it is also possible for more performant embeddings to diverge from human perceptions of similarity. As discussed, alignment between model vs. human embeddings can also be directly optimized (Liu et al., 2023; Toneva and Wehbe, 2019).

Of course, this, too, has a risk: tuning embeddings for perceptual judgments could improve explanation quality for users but reduce performance. Thus, the choice of embeddings may embody a tradeoff between performance and explanation quality, though current evidence suggests otherwise (Liu et al., 2023; Muttenthaler et al., 2023).

E When are Example-based Explanations Most Appropriate to Use?

Different input formats (e.g., text versus image) or categories (e.g., tweets vs. passages) impose varying amounts of cognitive load required to process and reason about analogical justifications. For example, the amount of critical thinking necessary to comprehend social media posts compared to scientific papers is drastically disparate, and that difference may even vary amongst users.

Consequently, user cognitive load in understanding example-based explanations is likely positively correlated with the amount of information inherently embedded in the inputs themselves. This directly impacts our analysis of explanation simplicity. For tweets, perhaps three or five short posts are still manageable. For scientific passages, maybe even one manuscript is overwhelming.

If model explanations are intended to support people, quantifying the degree to which explanations actually improve human performance in practice will ultimately require user studies.

F Visualizing L-Means

Figure 2 visualizes L-Means (described in Section 4) clusters for the training split of TOXIGEN and E-SNLI after using Principal Components Analysis (Maćkiewicz and Ratajczak, 1993) to reduce dimensions of extracted representations. Given limited space, only representations from DEBERTA-large, the top-performing model, are used.

F.1 TOXIGEN

For TOXIGEN, we observe an almost clean clustering of examples. Specifically, cluster 1 (crosses) consists mostly of toxic examples (1671 orange), with only 129 benign (blue) instances. Similarly, cluster 0 (circles) consists mostly of benign (5154 blue) examples, with only 26 toxic (orange) examples. As expected, the two clusters mainly separate along the first principal component, which accounts for almost half of the variation in DEBERTA encodings for TOXIGEN.

F.2 E-SNLI

We observe that the resultant clusters for E-SNLI are not as clean as those for TOXIGEN, hence resulting in noisier predictions that lead to slightly worse performance as shown in Table 3. This may be attributed to the fact that both principal components (PCs) constitute a significant amount of variation for E-SNLI, whereas PC1 for TOXIGEN is the sole dominant axis. Specifically, cluster 0 (circles) consists of 1168 entailment, 3996 neural, and 164217 contradiction examples. Cluster 1 contains 163618 entailment, 14229 neural, and 1437 contradiction examples. Cluster 2 comprises of 8628 entailment, 164537 neural, and 17531 contradiction examples.

G Qualitative Examples

We closely examine some qualitative example-based explanations for DEBERTA-large on the validation splits of TOXIGEN and E-SNLI. For the sake of space, we present the single closest (as measured by Euclidean distance over the latent representation space) neighbor/support vector/leaf node instance/cluster point as example-based explanations for various wrapper boxes. Instances are presented as is without any modifications, except that target groups and countries of offensive examples are demarcated in angle brackets. Raw

	Metric	BART vs. DeBERTa	BART vs. Flan-T5	DeBERTa vs. Flan-T5
ToxiGen	Accuracy	0.282	0.768	0.434
	Precision	0.192	**0.0248**	0.347
	Recall	0.0614	**<1e-3**	**<1e-3**
	F1	0.116	0.338	**0.0114**
E-SNLI	Accuracy	**<1e-3**	0.171	**0.024**
	Precision	**<1e-3**	0.188	**0.0112**
	Recall	**<1e-3**	0.180	**0.0203**
	F1	**<1e-3**	0.187	**0.0168**

Table 8: ToxiGen and E-SNLI test set p-values for comparisons between baseline transformers in Table 3

		BART-large				**DeBERTa-large**				**Flan-T5-large**			
		Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
ToxiGen	KNN	0.679	0.145	0.101	0.905	0.578	**0.014**	**0.007**	0.830	1.000	0.730	0.528	0.819
	DT	0.860	**0.016**	**<1e-3**	0.171	0.903	**0.006**	**<1e-3**	0.054	0.953	0.899	0.638	0.815
	L-Means	0.601	0.355	1.000	0.623	0.762	0.757	0.028	0.355	0.906	0.983	0.638	0.770
E-SNLI	KNN	0.791	0.740	0.777	0.762	0.054	**0.036**	**0.049**	**0.047**	0.137	0.142	0.144	0.145
	DT	**<1e-3**	**<1e-3**	**<1e-3**	**<1e-3**	0.796	0.762	0.758	0.774	0.843	0.831	0.841	0.832
	L-Means	**<1e-3**	**<1e-3**	**<1e-3**	**<1e-3**	**0.035**	0.259	**0.039**	0.053	0.767	0.755	0.764	0.756

Table 9: ToxiGen and E-SNLI test set p-values for Table 3.

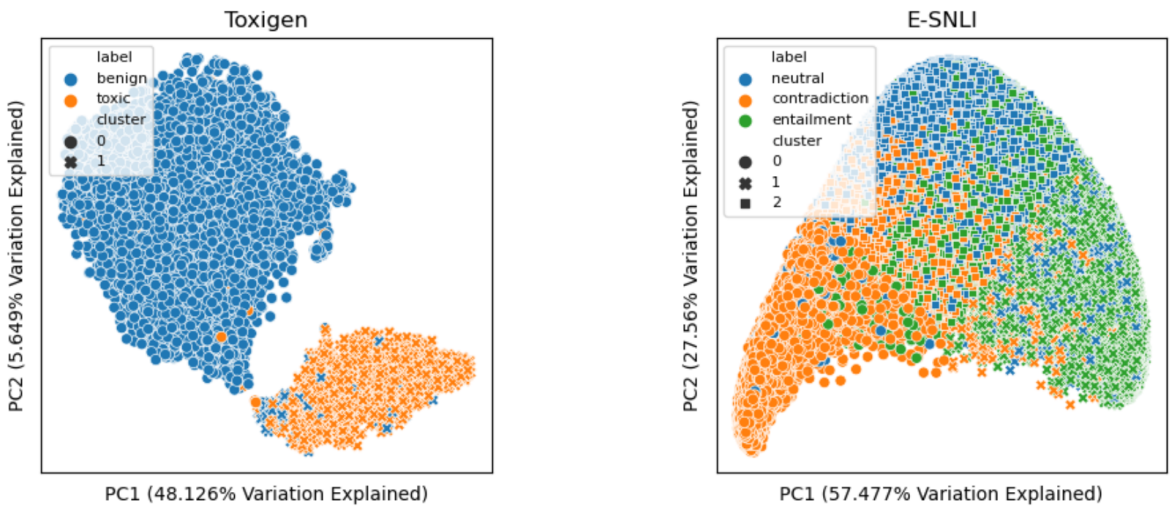

Figure 2: Visualization of resultant clusters from L-Means after PCA with two components.

Input	Classifier	Prediction	Explanation
people who are mentally disabled should be treated with the same respect and dignity that we would give to any other human being [Benign]	kNN	Benign	people who are mentally disabled should be treated with the same respect and dignity as anyone else [Benign]
	DT	Benign	Same as KNN
	L-means	Benign	Same as KNN
<GROUP> are very hostile people [Harmful]	kNN	Harmful	<GROUP> people arent normal [Harmful]
	DT	Harmful	Same as KNN
	L-means	Harmful	Same as KNN
the government should not be in the business of providing services for those who cannot help themselves [Harmful]	kNN	Benign	society should not be obligated to provide for the needs of those who are not able to provide for themselves [Harmful]
	DT	Harmful	it is a waste of money to provide services for those who are not productive members of the community [Harmful]
	L-means	Harmful	Same as KNN

Table 10: Qualitative examples for TOXIGEN validation set. Often, kNN, DT, and L-means identify the same closest training point as explanations. However, sometimes (row 3) wrapper boxes can lead to diverse explanations, and the final prediction may not agree with the label of the closest training point, since these explanations are parsimonious but not faithful. Nevertheless, we believe that these explanations are still useful, serving as intuitions for correct predictions and clarifications for failure cases.

examples may contain punctual, spelling, or grammatical errors. Labels for each training instance are in normal brackets.

Only presenting the closest training point does not constitute a faithful explanation of the wrapper boxes' reasoning process. Here, we again rely on the assumption that relevant examples as judged by the wrapper boxes will also be judged as related by users. Following this logic, we think the closest training point as deemed by the model is likely a good analogy to the input example from the users' perspective as well. However, the closest training point may not be representative of the overall distribution of the training instances applied for inference (e.g. row 3 in Table 10). Even if it may be relevant to the input, the closest example could be an outlier or an atypical example that does not accurately represent the majority of examples employed for reasoning. If users believe that explanations are faithful when they are not, this misinterpretation may also trick users into trusting faulty models (Lakkaraju and Bastani, 2020).

G.1 TOXIGEN

Table 10 showcases qualitative examples for TOXIGEN. Although not faithful, we observe that these explanations are relevant to the input text and are often identical across wrapper boxes. Specifically, kNN, DT, and L-means usually pinpoint the same training instances as explanations. Rows 1-2 illustrate this phenomenon, where all example-based explanations address the same topic as the inputs.

Since all wrapper boxes leverage more than just the closest training example in inference (Section 4), these explanations are simple but are not faithful (Case II in Table 2). This can lead to scenarios (row 3) where the final prediction disagrees with the explanation label. Furthermore, there's no guarantee that kNN, DT, and L-means always pinpoint the same explanations, and indeed they can be different since the exact mechanism by which similar examples are identified for each approach varies. Either way, we theorize that these explanations are useful to cultivate intuitions for correct predictions and clarifications for failure cases.

G.2 E-SNLI

Table 11 presents qualitative examples for E-SNLI. Each sample constitutes a premise-hypothesis pair, alongside a randomly selected (from three) human-annotated explanation. Although our qualitative example-based explanations (Table 10 and Table 11) are simple and intuitive, other NLP tasks may differ, such as topic modeling or passage re-

Input	Classifier	Prediction	Explanation
Premise: Two women are embracing while holding to go packages. Hypothesis: Two woman are holding packages. [Entailment] Human explanation: Saying the two women are holding packages is a way to paraphrase that the packages they are holding are to go packages.	kNN	Entailment	Premise: Two boys show off their stained, blue tongues. Hypothesis: boys are showing their tongues. [Entailment]
	DT	Entailment	Same as KNN
	L-means	Entailment	Premise: This young child is having fun on their first downhill sled ride. Hypothesis: A child on a sled. [Entailment]
Premise: A shirtless man is singing into a microphone while a woman next to him plays an accordion. Hypothesis: He is playing a saxophone. [Contradiction] Human explanation: A person cannot be singing and playing a saxophone simultaneously.	kNN	Contradiction	Premise: A woman is sitting on a steps outdoors playing an accordion. Hypothesis: Someone is playing a piano. [Contradiction]
	DT	Contradiction	Same as KNN
	L-means	Contradiction	Premise: Africans gather water at an outdoor tap. Hypothesis: Africans are gathering rice for a meal. [Contradiction]
Premise: A woman in a gray shirt working on papers at her desk. Hypothesis: Young lady busy with her work in office. [Neutral] Human explanation: All women are not young. Although she is working on papers at her desk, it does not mean that she is busy or that she's in an office.	kNN	Neutral	Premise: Man raising young boy into the clear blue sky. Hypothesis: Father holds his son in the air. [Entailment]
	DT	Entailment	Premise: Two soccer players race each other during a match while the crowd excitedly cheers on. Hypothesis: Two men compete to see who is faster during soccer. [Entailment]
	L-means	Neutral	Premise: A model poses for a photo shoot inside a luxurious setting. Hypothesis: a woman poses. [Neutral]

Table 11: Qualitative examples for E-SNLI validation set. Here, most of the time, kNN and DT identify the same closest training point as explanations, and L-means pinpoints a different but related instance (rows 1-2). We postulate that this differs from TOXIGEN because L-means results in less clean clusters. Again, sometimes (row 3) wrapper boxes can still lead to diverse explanations, and the final prediction may not agree with the label of the closest training point. Nevertheless, we believe that these explanations are still useful as rationales behind annotated human explanations often also apply to the selected training examples.

trieval.

Interestingly, whereas for TOXIGEN kNN, DT, and L-means often pinpoint the same training instances as explanations, for E-SNLI instead kNN and DT follow this trend (rows 2-3). We postulate that this occurs because L-means results in less clean clusters (noisier neighbors)

Nevertheless, we observe that provided example-based explanations often require the same reasoning skills as the input, consistent with examples from TOXIGEN. Again, sometimes (row 3) wrapper boxes can still lead to different explanations, and the final prediction may not agree with the label of the closest training point.

Unfaithful explanations can still be useful. For E-SNLI, we observe that rationales behind annotated human explanations often apply to the presented example-based explanations. For example, the human justifies the pair as entailment for the first input example (row 1) since the hypothesis paraphrases the premise. Likewise, our example-based explanation displays a hypothesis that paraphrases the premise.

H Wrapper Box Results for LLMs

H.1 Evaluation Setup

We additionally experiment with several more modern large language models (LLMs) to test the generalizability of wrapper boxes. Namely, we experiment with Llama 2-7B Instruct (Touvron et al., 2023), Llama 3-8B-Instruct (Dubey et al., 2024), Mistral-7B Instruct (Jiang et al., 2023), and Gemma-7B Instruct (Mesnard et al., 2024). We used model checkpoints publicly hosted on Hugging Face. Table 14 shows the prompts used for all LLMs.

Representations are extracted from the penultimate layer (directly preceding the language modeling heads) and mean-pooled across tokens to obtain a sentence-level embedding for white classic models. Sentence representations are then further processed by fitting a logistic regression, and we then take the logit output of the regression model to be the final input features for wrapper boxes. We empirically observe that this logistic transformation is necessary to achieve comparable performance and that fitting wrapper boxes directly on mean-pooled embeddings can lead to severe performance degradation, particularly for L-means.

We use the same datasets and metrics introduced in Section 5. Due to time and computation constraints, we only report zero-shot results and only report metrics on a 10,000 random stratified sample for E-SNLI. Applying state-of-the-art in-context-learning (ICL) strategies or fine-tuning may improve baseline neural performance but would also result in different representations as input to wrapper boxes. The efficacy of wrapper boxes in these scenarios would thus need to be empirically benchmarked, although we anticipate that there should be no drastic performance degradation.

H.2 TOXIGEN LLM Results

Table 12 shows predictive performance results for TOXIGEN. LLM zero-shot results favor recall over precision, likely due to the imbalanced distribution of labels in TOXIGEN, with Llama 3 8B being the best model. No wrapper box uniformly outperforms others, although the precision and recall scores are more balanced. Results show that wrapper boxes using zero-shot LLM representations strongly outperform baseline LLM performance across both tasks.

H.3 E-SNLI LLM Results

Table 13 shows predictive performance results for E-SNLI. Precision and recall scores are balanced for both LLMs and wrapper boxes here since E-SNLI has a balanced distribution of labels. Interestingly, we observe that decision tree consistently outperforms other wrapper boxes, although the differences (e.g., compared to KNN) may not be significant. Nevertheless, we consistently observe that wrapper boxes can strongly outperform baseline LLMs using zero-shot LLM representations.

Model	Classifier	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
LLama2 7B	Zero-shot	61.17	55.76	42.51	**80.99**
	KNN	82.02	68.05	**73.47**	63.38
	DT	80.85	69.90	66.56	73.59
	L-Means	**82.23**	**71.06**	69.97	72.18
LLama3 8B	Zero-shot	71.70	**65.81**	51.82	**90.14**
	KNN	**77.23**	59.47	**64.34**	55.28
	DT	76.70	61.10	61.65	60.56
	L-Means	76.60	61.27	61.27	61.27
Mistral 7B	Zero-shot	68.62	**64.50**	48.99	**94.37**
	KNN	78.72	62.69	66.67	59.15
	DT	**79.04**	63.59	**66.93**	60.56
	L-Means	78.72	63.90	65.56	62.32
Gemma 7B	Zero-shot	55.64	56.79	40.23	**96.48**
	KNN	79.36	63.12	**68.60**	58.45
	DT	78.94	**68.37**	62.57	75.35
	L-Means	**79.47**	66.55	65.53	67.61

Table 12: TOXIGEN test set performance metrics (percentage) of wrapper boxes for various instruction-tuned, open-source large language models. Embeddings are mean-pooled overall all sequence tokens from the penultimate layer (directly preceding the generative language modeling head). Then, representations are further transformed into logits using weights learned from logistic regressor before input as features to other wrapper boxes (KNN, DT, L-Means).

Model	Classifier	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
LLama2 7B	Zero-shot	47.96	42.28	50.52	48.51
	KNN	70.80	70.76	70.92	70.72
	DT	**73.03**	**72.91**	**72.91**	**72.96**
	L-Means	65.95	66.42	69.37	65.87
LLama3 8B	Zero-shot	61.93	62.71	68.48	61.89
	KNN	70.75	70.72	70.86	70.67
	DT	**73.35**	**73.31**	**73.34**	**73.30**
	L-Means	66.43	66.83	69.53	66.33
Mistral 7B	Zero-shot	70.99	71.47	72.81	70.99
	KNN	80.23	80.15	80.13	80.18
	DT	**81.94**	**81.90**	**81.91**	**81.91**
	L-Means	77.20	76.27	76.59	77.00
Gemma 7B	Zero-shot	52.56	48.12	70.30	53.20
	KNN	66.07	65.98	66.15	65.97
	DT	**69.68**	**69.68**	**69.80**	**69.62**
	L-Means	62.61	63.02	66.41	62.50

Table 13: E-SNLI test set performance metrics (percentage) of wrapper boxes for various instruction-tuned, open-source large language models. Embeddings are mean-pooled overall all sequence tokens from the penultimate layer (directly preceding the generative language modeling head). Then, representations are further transformed into logits using weights learned from logistic regressor before input as features to other wrapper boxes (e.g., KNN, DT, L-Means). The training set is a 10,000 random stratified sample of the original 500k set.

Dataset	Prompt
TOXIGEN	You are a content moderator tasked with toxicity detection. After reading the provided text, categorize it as either "0" for benign or "1" for toxic. Only respond with "0" or "1". Do not include any additional text or explanations.
E-SNLI	You are an informative AI assistant tasked with natural language inference. Given the following premise and hypothesis, classify their relationship as "0" for entailment, "1" for neutral, or "2" for contradiction. **Entailment (0)**: The hypothesis logically follows from the premise. If the premise is true, the hypothesis must also be true. **Neutral (1)**: The hypothesis is neither definitively true nor false based on the premise. The premise provides some information, but it is insufficient to confirm or deny the hypothesis. **Contradiction (2)**: The hypothesis directly conflicts with the premise. If the premise is true, the hypothesis must be false. Only respond with "0", "1", or "2". Do not include any additional text or explanations.

Table 14: LLM Prompts for TOXIGEN and E-SNLI.