Can Explanations Be Useful for Calibrating Black Box Models?

Anonymous ACL submission

Abstract

One often wants to take an existing, trained NLP model and use it on data from a new domain. While fine-tuning or few-shot learning can be used to adapt the base model, there is no one simple recipe to getting these working; moreover, one may not have access to the original model weights if it is deployed as a black box. To this end, we study how to improve a black box model's performance on a new domain given examples from the new domain by leveraging *explanations* of the model's behavior. Our approach first extracts a set of features combining human intuition about the task with model attributions generated by black box interpretation techniques, and then uses a simple model to calibrate or rerank the model's predictions based on the features. We experiment with our method on two tasks, extractive question answering and natural language inference, covering adaptation from several pairs of domains. The experimental results across all the domain pairs show that explanations are useful for calibrating these models. We show that the calibration features transfer to some extent between tasks and shed light on how to effectively use them.

1 Introduction

With recent breakthroughs in pre-trained modeling, NLP models are showing increasingly promising performance on real-world tasks, leading to their deployment at scale for settings such as translation, sentiment analysis, and question answering. These models are sometimes used as black boxes, especially if they are only available as a service through APIs[1] or if end users do not have the resources to fine-tune the models themselves. This poses a challenge when users try to deploy models on a new domain that diverges from the training domain, usually resulting in performance deterioration.

To this end, we investigate the task of domain adaptation of black box models: given a black box model and a small number of examples from a new domain, how can we improve the model's generalization performance on the new domain? In this setting, we are not able to update the model parameters, which makes transfer and few-shot learning techniques inapplicable. Furthermore, we cannot even *access* the model parameters, ruling out techniques requiring model internal representations.

This paper explores how explanations can help address this task. We leverage black box feature attribution techniques (Ribeiro et al., 2016; Lundberg and Lee, 2017) to interpret a model's internal reasoning process. As shown in Figure 1, we use this knowledge in a *calibrator*, or a separate model to make a binary decision of whether the black box model is likely to be correct or not on a given instance. While not fully addressing the domain adaptation problem, calibrating the model can make it more useful in practice, as we can recognize when it is likely to make mistakes (Guo et al., 2017; Kamath et al., 2020; Desai and Durrett, 2020) and modify our deployment strategy accordingly.

We calibrate by connecting model interpretations with hand-crafted heuristics to extract a set of features describing the reasoning of the model. Figure 1 shows an example for question answering: we believe the answers are more reliable when the tokens of a particular set of tags (e.g., proper nouns) in the question are strongly considered. We extract a set of features describing the attribution values of different tags. Using a small number of examples in the target domain, we can train a simple calibrator for the black box model.

Our approach is closely related to the recent line of work on model behavior and explanations. Chandrasekaran et al. (2018); Hase and Bansal (2020) shows explanations can help users predict model

[1] Google Translate, the Perspective API https://perspectiveapi.com/, and MonkeyLearn https://monkeylearn.com/monkeylearn-api/ being three examples.

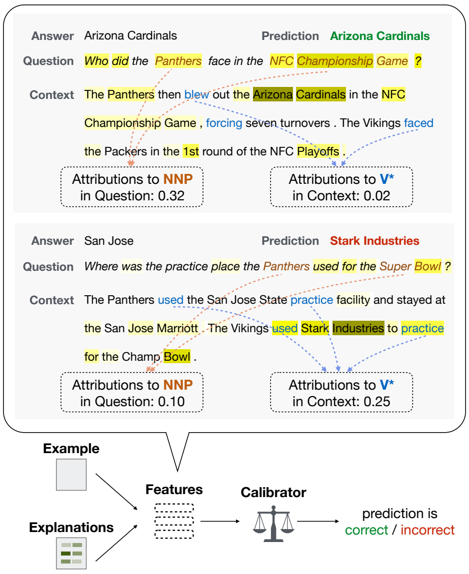

Figure 1: Pipeline and examples from the SQuAD-ADV dataset. A RoBERTa model trained on SQuAD resists the attack on the first example but fails on the second. Features that inspect attribution values produced by LIME can differentiate these two on the basis of attributions to NNP in the question and V* in the context. A calibrator can use these features to predict whether the original black box model was right or wrong.

decisions in some ways and Ye et al. (2021) show how these explanations can be semi-automatically connected to model behavior. Our approach goes further by using a model to learn these heuristics, instead of handcrafting them or having a human inspect the explanations.

We test whether our method can improve model generalization performance on two tasks, extractive question answering (QA) and natural language inference (NLI). We construct generalization tasks for 5 pairs of source and target domains across the two tasks. Compared to existing baselines (Kamath et al., 2020) and our own ablations, we find explanations are indeed helpful for this task, successfully improving model generalization performance among all pairs. Although the number of examples needed for training a calibrator is sometimes sufficient to adapt a trained model, we still find occasions where explanation-based calibrators outperform even methods that have full access to the models. Our analysis demonstrates promising cross-domain generalization ability of explanation-based calibrators: our calibrator trained on a new domain can transfer to another new domain in some cases. Moreover, our calibrator can also substantially improves model performance in the Selective QA setting.

2 Using Explanations for Black Box Model Calibration

Let $x = x_1, x_2, ..., x_n$ be a set of input tokens and $\hat{y} = f(x)$ be a prediction from our black box model under consideration. Our task in calibration[2] is to **assess whether the model prediction on** x **matches its ground truth** y. We represent this with the variable t, i.e., $t \triangleq \mathbb{1}\{f(x) = y\}$.

We explore various calibrator models to perform this task, with our main focus being on calibrator models that leverage explanations in the form of *feature attribution*. Specifically, an explanation ϕ for the input x assigns an attribution score ϕ_i for each input token x_i, which represents the importance of that token. Next, we extract features $u(x, \phi)$ depending on the input and explanation, and use the features to learn a calibrator $c : u(x, \phi) \rightarrow t$ for predicting whether a prediction is valid. We compare against baselines that do not use explanations in order to answer the core question posed by our paper's title.

Our evaluation focuses on binary calibration, or classifying whether a model's initial prediction is correct. Following recent work in this setting Kamath et al. (2020), we particularly focus on domain transfer settings where models make frequent mistakes. A good calibrator can identify instances where the model has likely made a mistake, so we can return a null response to the user instead of an incorrect one.

In the remainder of this section, we'll first introduce how we generate the explanations and then how to extract the features u for the input x.

2.1 Generating Explanations

Since we are calibrating black box models, we adopt LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) for generating explanations for models instead of other techniques that require access to the model details (e.g., integrated gradients (Sundararajan et al., 2017)).

The rest of this work only relies on LIME and SHAP to map an input sequence x and a model prediction y to a set of importance weights ϕ. We will briefly summarize the unified framework shared by

[2]We follow Kamath et al. (2020) in treating calibration as a binary classification task. Devising a good classifier is connected to the goal of accurate estimation of posterior probabilities that calibration has more historically referred to (Guo et al., 2017), but our evaluation focuses on binary accuracy rather than real-valued probabilities.

both methods, and refer readers to the respective papers for additional details.

LIME and SHAP generate *local explanations* by approximating the model's predictions on a set of perturbations around the base data point x. In this setting, a perturbation x' with respect to x is a simplified input where some of the input tokens are absent (replaced with a `<mask>` token). Let $z = z_1, z_2, ..., z_n$ be a binary vector with each z_i indicating whether x_i is present (using value 1) or absent (using value 0), and $h_x(z)$ be the function that maps z back to the simplified input x'. Both methods seek to learn a local linear classifier g on z which matches the prediction of original model f by minimizing:

$$g(z) = \phi_0 + \sum_{i=1}^{n} \phi_i z_i$$

$$\xi = \arg\min_{g} \sum_{z \in Z} \pi_x(z)[f(h_x(z)) - g(z)]^2 + \Omega(g)$$

where π_x is a local kernel assigning weight to each perturbation z, and Ω is the L2 regularizer over the model complexity. The learned feature weight ϕ_i for each z_i then represents the additive attribution (Lundberg and Lee, 2017) of each individual token x_i. LIME and SHAP differ in the choice of the local kernel π_x. Please refer to the supplementary materials for details of the kernel.

2.2 Extracting Features by Combining Explanations and Heuristics

Armed with these explanations, we now wish to connect the explanations to *the reasoning we expect from the task*: if the model is behaving as we expect it, it may be better calibrated. A human might look at the attributions of some important features and decide whether the model is trustworthy in a similar fashion (Doshi-Velez and Kim, 2017). Past work has explored such a technique to compare explanation techniques (Ye et al., 2021), or even used actual human users to do this task (Chandrasekaran et al., 2018; Hase and Bansal, 2020).

Our method automates this process by learning what properties of explanations are important. We first assign each token x_i with one or more human-understandable **properties** $V(x_i) = \{v_j\}_{j=1}^{m_i}$. Each property $v_j \in \mathcal{V}$ is an element in the property space, which includes indicators like POS tags and is used to describe an aspect of x_i whose importance might correlate with the model's robustness. We intend to conjoin these properties with aspects of the explanation to render our calibration judgment. Figure 1 shows examples of properties such as whether a token is a proper noun (NNP).

We now construct the feature set for the prediction made on x. For every property $v \in \mathcal{V}$, we extract a single feature $F(v, x, \phi)$ by aggregating the attributions of the tokens associated with v:

$$F(v, x, \phi) = \sum_{i=1}^{n} \sum_{\bar{v} \in V(x_i)} \mathbb{1}\{\bar{v} = v\}\phi_i$$

where $\mathbb{1}$ is the indicator function, and ϕ_i is the attribution value. In this way, an individual feature represents the total attributions with respect to property v when the model is making the predictions for x. The complete feature set u for x, given as $u = \{F(v, x, \phi)\}_{v \in \mathcal{V}}$, can summarize model rationales from the perspective of the properties defined in $\mathcal{V}$.

Properties We use several types of heuristic properties for calibrating QA and NLI models.

Segments of the Input (QA and NLI): In both of our tasks, an input sequence can naturally be decomposed into two parts, namely a question and a context (QA) or a premise and a hypothesis (NLI). We assign each token with the corresponding segment name, which yields features like `Attributions to Question`.

POS Tags (QA and NLI): We also use tags from the English Penn Treebank (Marcus et al., 1993) to implement a group of properties. We hypothesize that tokens of some specific tags should be more important, like proper nouns in the questions of the QA tasks. If a model fails to consider proper nouns of a QA pair, it is more likely to make incorrect predictions.

Overlapping Words (NLI): Word overlapping strongly affects model prediction (McCoy et al., 2019). We assign each token with a property of `Overlapping` or `Non-Overlapping`.

Conjunction of Groups: We can further produce higher-level properties by taking the Cartesian product of two or more groups. We conjoin `Segment` and `Pos-Tags`, which yields higher-level features like `Attributions to NNP in Question`. Such a feature aggregates attributions of tokens that are tagged with `NNP` and also required to be in the question (marked with orange).

2.3 Calibrator Model

We train the calibrator on a small number of samples in our target domain. Each sample is labeled using the prediction of the original model compared to the ground truth. Using our feature

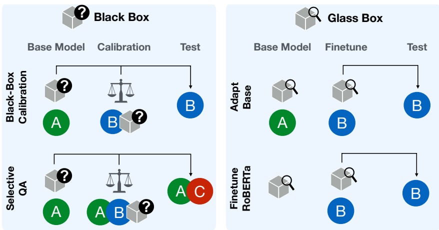

Figure 2: Illustration of different settings in the experiments. In black box settings, a calibrator is trained for improving model performance on OOD data; in glass box settings, the model is finetuned on OOD data from a base model or vanilla RoBERTa LM model.

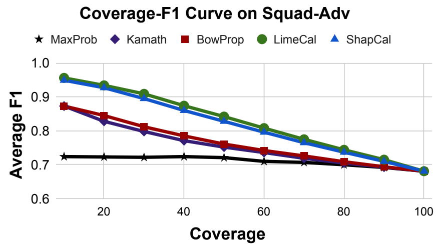

Figure 3: Coverage-F1 curves of different approaches on SQuAD-ADV. As more low-confidence questions are answered, the average F1 scores decrease. We use AUC to evaluate calibration performance.

set $F(v, x, \phi)$, we learn a random forest classifier, shown to be effective for a similar data-limited setting in Kamath et al. (2020), to predict t (whether the prediction is correct). This classifier returns a score, which overrides the model's original confidence score with respect to that prediction.

In Section 4, we discuss several baselines for our approach. Whenever we vary the features used by the model, all the other details of the classifier and setup remain the same.

3 Tasks and Datasets

Our task setup involves transferring from a source domain/task A to a target domain/task B. Figure 2 shows the data condition we operate in. Our primary experiments focus on using our features to either calibrate or selectively answer in the black box setting (top-left in Figure 2). In this setting, we have a black box model trained on an source domain A and a small amount of data from the target domain B. Our task is to train a calibrator using data from domain B to identify instances where the model potentially fails in the large unseen test data in domain B. We contrast this black box setting with glass box settings (right column in Figure 2). In glass box settings, we directly have access to the model parameters, making it possible to finetune a model on domain B or train on B from scratch.

Question Answering We experiment with domain transfer from SQuAD (Rajpurkar et al., 2016) to three different settings: SQuAD-ADV (Jia and Liang, 2017), HOTPOTQA (Yang et al., 2018), and TRIVIAQA (Joshi et al., 2017).

SQuAD-ADV is an adversarial setting based on SQuAD, which constructs adversarial QA examples based on SQuAD by appending a distractor sentence at the end of each example's context. The added sentence contains a spurious answer and usually has high surface overlapping with the question so as to fool the model. We use the ADDSENT setting from Jia and Liang (2017).

Similar to SQuAD, HOTPOTQA also contains passages extracted from Wikipedia, but HOTPOTQA asks questions requiring multiple reasoning steps. TRIVIAQA is collected from Web snippets, which present a different distribution of questions and passages than SQuAD. For HOTPOTQA and TRIVIAQA, we directly use the pre-processed version of dataset from the MRQA Shared Task (Fisch et al., 2019).

NLI For the task of NLI, we transfer a model trained on MNLI (Williams et al., 2018) to MRPC (Dolan and Brockett, 2005) and QNLI (Wang et al., 2019), similar to the settings in Ma et al. (2019). QNLI contains a question and context sentence pair from SQuAD, and the task is to verify whether a sentence contains the answer to the paired question. MRPC is a paraphrase detection dataset presenting a binary classification task to decide whether two sentences are paraphrases of one another. Note that generalization from MNLI to QNLI or MRPC not only introduces shift in terms of the distribution of the input text, but in terms of the nature of the task itself, since QNLI and MRPC aren't strictly NLI tasks despite sharing some similarity. Both are *binary* classification tasks rather than three-way.

4 Experiments

Baselines We compare our calibrator against existing baselines as well as our own ablations.

MAXPROB simply uses the probability of the top prediction to assess whether the prediction is trustworthy.

KAMATH (Kamath et al., 2020) (for QA only) is a baseline initially proposed to distinguish out-of-distribution data points from in-domain data points in the SELECTIVE QA setting, but it can also be applied in our settings. It trains a random forest classifier to learn whether a model's prediction is correct based on several heuristic features, including the probabilities of the top 5 predictions, the length of the context, and the length of the predicted answer. Since we are calibrating black box models, we do not use dropout-based features in Kamath et al. (2020).

CLSPROB (for NLI only) uses more detailed information than MAXPROB: it uses the predicted probability for `Entailment`, `Contradiction`, and `Neutral` as the features for training a calibrator instead of only using the maximum probability.

BOWPROP adds a set of heuristic property features on top of the KAMATH method. These are the same as the features used by the full model *excluding the explanations*. We use this baseline to give a baseline for using general "shape" features on the inputs *not* paired with explanations.

Implementation of Our Method We refer our explanation-based calibration method using explanations produced by LIME and SHAP as **LIMECAL** and **SHAPCAL** respectively. We note that these methods also take advantages of the bag-of-word features in BOWPROP. For QA, the property space is the union of low-level `Segment` and `Segment` × `Pos-Tags`. For NLI, we use the union of `Segment` and `Segment` × `Pos-Tags` × `Overlapping Words` to label the tokens. Details number of features can be found in the Appendix.

4.1 Main Results: QA

Setup We train a ROBERTA (Liu et al., 2019) QA model on SQUAD as the base model, which achieves 85.5 exact match and 92.2 F1 score. For the experiments on HOTPOTQA and TRIVIAQA, we split the dev set and sample 500 examples for training, and the rest for testing.[3] For experiments on SQUAD-ADV, we remove the unmodified data points in the ADD-SENT setting and also use 500 examples for training. For the experiments across all pairs, we randomly generate the splits, test the methods 20 times, and average the results to alleviate the influence of randomness.

Metrics In addition to *calibration accuracy* (**ACC**) that measures the accuracy of the calibrator, we also use the *area under coverage-F1 curve* (**AUC**) to evaluate the calibration performance for QA tasks in particular. The coverage-F1 curve (Figure 3) plots the average F1 score of the model achieved when the model only chooses to answer varying fractions (coverage) of the examples ranked by the calibrator-produced confidence. A better calibrator should assign higher scores to the questions that the models are sure of, thus resulting in higher area under the curve; note that AUC of 100 is impossible since the F1 is always bounded by the base model when every question is answered. We additionally report the average scores when answering the top 25%, 50%, and 75% questions, for a more intuitive comparison of the performance.

Results Table 1 summarizes the results for QA. First, we show that using explanations are helpful for calibrating black box QA models out-of-domain. Our method using LIME substantially improves the calibration AUC compared to KAMATH by 7.1, 2.1 and 1.4 on SQUAD-ADV, TRIVIAQA, and HOTPOTQA, respectively. In particular, LIMECAL achieves an average F1 score of 92.3 at a coverage of 25% on SQUAD-ADV, close to the performance the base model on original SQUAD examples. Our explanation-based approach is effective at identifying the examples that are robust with respect to the adversarial attacks.

Comparing LIMECAL against BOWPROP, we find that the explanations themselves do indeed help. On SQUAD-ADV and HOTPOTQA, BOWPROP performs on par with or only slightly better than KAMATH. These results show that connecting explanations with annotations is a path towards building better calibrators.

Finally, we compare the performance of our methods based on different explanation techniques. LIMECAL slightly outperforms SHAPCAL in all three settings. As discussed in Section 2.1, SHAP assigns high instance weights to those perturbations with few activated features. While such a choice of the kernel is effective in tasks involving tabular data (Lundberg and Lee, 2017), this might not be appropriate for the task of QA when such perturbations may not yield meaningful examples.

4.2 Main Results: NLI

Setup Our base NLI model is a ROBERTA classification model trained on MNLI and achieves 87.7% accuracy on the development set. We collapse `contradiction` and `neutral` into `non-entailment` when evaluating on QNLI

[3]Details of hyperparameters can be found in the Appendix.

Approach	SQUAD-ADV Acc	AUC	F1@25	F1@50	F1@75	TRIVIAQA Acc	AUC	F1@25	F1@50	F1@75	HOTPOTQA Acc	AUC	F1@25	F1@50	F1@75
MAXPROB	62.6	70.9	72.4	72.1	70.4	67.0	76.7	82.1	76.3	71.0	63.1	75.7	79.7	75.9	72.2
KAMATH	63.2	76.8	81.4	75.2	71.2	70.6	76.6	82.1	77.9	71.1	64.5	76.8	80.8	77.2	72.8
BOWPROP	63.6	77.4	82.9	76.1	71.7	71.2	77.6	84.2	79.1	71.6	64.7	76.6	80.3	76.9	72.4
LIMECAL	**70.3**	**83.9**	**92.3**	**84.2**	**75.9**	**72.0**	**78.7**	**85.4**	**79.6**	**72.3**	**65.7**	**78.2**	**82.6**	**78.4**	**73.8**
SHAPCAL	69.3	82.9	91.2	82.8	75.0	71.8	78.2	84.7	79.4	72.3	65.3	77.8	82.0	78.0	73.5

Table 1: Main results on QA tasks. Our explanation-based methods (LIMECAL and SHAPCAL) successfully calibrate a ROBERTA QA model trained on SQUAD when transferring to three new domains, and outperform a prior approach (KAMATH) as well as our ablation using only heuristic labels (BOWPROP).

Approach	QNLI Acc	QNLI AUC	MRPC Acc	MRPC AUC
MAXPROB	50.5	41.2	57.0	50.0
CLSPROB	56.7	59.5	71.5	77.9
BOWPROP	74.0	82.0	71.8	79.3
LIMECAL	**75.0**	**82.6**	**73.6**	**81.0**
SHAPCAL	74.2	81.9	73.5	80.7

Table 2: Main results on NLI tasks. LIMECAL moderately improves the performance of the base MNLI model on QNLI and MRPC, despite how different these tasks are from the base MNLI setting.

and MRPC. We also use random forests as the calibrator model. We evaluate the generalization performance on the development sets of QNLI and MRPC. Similar to the settings in QA, we use 500 examples to train the calibrator and test on the rest for each of the 20 random trials.

Metrics Because QNLI and MRPC are binary classification tasks, predicting whether a model is correct (our calibration setting) is equivalent to the original prediction task. We can therefore measure calibrator performance with standard classification accuracy and AUC.

Results We show results on NLI tasks in Table 2. The base MNLI model utterly fails when transferring to QNLI and MRPC and achieves an accuracy of 49% and 57%, respectively, whereas the majority class is 50% (QNLI) and 65% (MRPC). With heuristic annotations, BOWPROP is able to solve 74% of the QNLI instances and 72% of the MRPC instances. Our heuristic itself is strong for QNLI compared to MAXPROB. LIMECAL is still the best in both settings, moderately improving accuracy by 1% and 2% over BOWPROP using explanations. The results on NLI tasks suggest our method can still learn useful signals for indicating model reliability even if the underlying tasks are very different.

4.3 Analysis

Cross-Domain Generalization of Calibrators Our calibrators so far are trained on individual transfer settings. Is the knowledge of a calibrator learned on some initial domain transfer setting, e.g., SQuAD $\rightarrow$ TRIVIAQA, generalizable to another transfer setting, e.g. $\rightarrow$ HOTPOTQA? This would enable us to take our basic QA model and a calibrator and apply *that pair* of models in a new domain without doing any new training or adaptation. We explore this hypothesis on QA.[4]

For comparison, we also give the performance a ROBERTA-model first finetuned on SQUAD and then finetuned on domain A (ADAPT, Figure 2). ADAPT requires access to the model architecture and is an unfair comparison for other approaches.

We show the results in Table 5. None of the approaches can generalize between SQUAD-ADV and the other domains (either trained or tested on SQUAD-ADV), which is unsurprising given the synthetic and very specific nature of SQUAD-ADV.

Between TRIVIAQA and HOTPOTQA, both the LIMECAL and KAMATH calibrators trained on one domain can generalize to the other, even though BOWPROP is not effective. Furthermore, our LIMECAL exhibits a stronger capability of generalization compared to KAMATH. We then compare LIMECAL against ADAPT. ADAPT does not always work well, which has also been discussed in Kamath et al. (2020); Talmor and Berant (2019). ADAPT leads to a huge drop in terms of performance when being trained on HOTPOTQA and tested on TRIVIAQA, whereas LIMECAL is the best in this setting. From TRIVIAQA to HOTPOTQA, ADAPT works well, but LIME is almost as effective.

Overall, the calibrator trained with explanations as features exhibits successful generalizability across the two realistic QA tasks. We believe this can be attributed to the features used in the explanation-based calibrator. Although the task is different, the calibrator can rely on some common rules to decide the reliability of a prediction.

[4] We also tested the hypothesis on the NLI-paraphrase transfer, but did not see evidence of transferability there, possibly due to the fact that these tasks fundamentally differ.

	SQuAD-Adv			TriviaQA			HotpotQA			QNLI			MRPC		
	100	300	500	100	300	500	100	300	500	100	300	500	100	300	500
MaxProb		70.9			76.7			75.7			41.2			50.0	
Kamath	72.7	75.6	76.8	74.8	76.2	76.6	75.2	76.5	76.8	56.4	58.1	59.5	73.7	76.8	77.9
BowProp	75.0	76.0	77.4	76.1	77.4	77.6	74.9	76.3	76.6	79.0	81.5	82.0	69.4	77.5	79.3
LimeCal	**78.7**	**82.7**	**83.9**	**77.2**	**78.2**	**78.7**	**76.5**	**77.7**	**78.2**	**79.1**	**81.8**	**82.8**	**76.1**	**79.9**	**81.0**

Table 3: AUC scores of the calibrators trained with varying training data size. Explanation-based calibrators can still learn even with limited training resource, whereas Kamath and BowProp are not effective and underperform the MaxProb baseline on TriviaQA and HotpotQA.

	SQuAD-Adv		TriviaQA		HotpotQA		QNLI	MRPC
Model Performance	Ex	F1	Ex	F1	Ex	F1	Acc	Acc
Base QA/NLI	62.1	68.0	53.2	62.1	50.7	66.3	50.5	57.2
Finetune RoBERTa	32.3	42.0	28.5	34.8	39.5	54.8	81.2	79.8
Adapt Base QA/NLI	77.3	84.3	56.2	64.0	54.3	70.8	80.7	79.1
InDomain QA/NLI	–	–	62.1	68.1	59.7	77.2	92.0	87.2
Calibration Results	Acc	AUC	Acc	AUC	Acc	AUC	Acc	Acc
Finetune RoBERTa + MaxProb	–	41.1	–	37.6	–	67.0	**81.2**	**79.8**
Adapt Base QA/NLI + MaxProb	–	**92.7**	–	77.6	–	**82.5**	80.7	79.1
LimeCal	69.3	82.9	72.0	**78.7**	65.7	78.2	74.9	73.6

Table 4: Model performance and calibration performance of LimeCal and glass box methods. On QA tasks, LimeCal is better than Finetuning RoBERTa and even outperforms Adapt Base QA/NLI on TriviaQA. LimeCal under-performs glass box methods on NLI due to its easy nature and the poor base-model performance.

Source \ Target		Sq-Adv	Trivia	Hotpot
Sq-Adv	Adapt	70.9	76.1	65.8
	Kamath		73.3	75.1
	BowProp		71.9	74.1
	LimeCal		72.9	71.4
Trivia	Adapt	64.2	76.7	**77.2**
	Kamath	70.5		76.7
	BowProp	67.1		75.0
	LimeCal	69.3		77.0
Hotpot	Adapt	56.6	74.0	75.7
	Kamath	70.6	77.0	
	BowProp	69.1	76.9	
	LimeCal	68.8	**77.9**	

Table 5: Area under Coverage-F1 curve for cross-domain calibration results. The numbers along the diagonal shows the MaxProb performance. A better performance than MaxProb suggests the calibrator is able to usefully generalize.

Impacts of Training Data Size Calibrating a model for a new domain becomes cumbersome if large amounts of annotated data are necessary. We experiment with varying the amount of training data the calibrator is exposed to, with results shown in Table 3. Our explanation-based calibrator is still the best in every setting with as few as 100 examples. With 100 examples, Kamath and BowProp perform worse than the MaxProb baseline on TriviaQA and HotpotQA, indicating that more data is needed to learn to use their features.

4.4 Comparison to Finetuned Models

Throughout this work, we have assumed a black box model that cannot be fine-tuned on a new domain. In this section, we compare calibration-based approaches with glass-box methods that require access to the model architectures and parameters. We evaluate two glass-box methods in two different settings (Figure 2): (1) finetuning a base RoBERTa model (Finetune RoBERTa), which needs the access to the model architectures but not parameters; and (2) finetuning a base QA/NLI model, which requires both model architectures as well as parameters. All these models are finetuned with 500 examples, the same as LimeCal. We also give the performance of a model trained with full in-domain training data for different tasks as references (InDomain QA/NLI).

We present the model performance (measured with Exact Match and F1 for QA and Acc for NLI) and calibration results in Table 4. Note that there are no calibrators for glass box methods, so we only report AUC scores for calibration performance.

On QA tasks, the limited training data is not sufficient for successfully finetuning a RoBERTa model. Consequently, Finetune RoBERTa does not achieve credible performance. Finetuning a base QA model greatly improves the performance, surpassing LimeCal on SQuAD-Adv and HotpotQA. However, we still find that on TriviaQA, LimeCal slightly outperforms Adapt. This is a surprising result, and shows that explanation-based calibrators can still be beneficial in some scenarios, even if we have full access to the model.

On NLI tasks that are substantially easier than QA, finetuning either a RoBERTa LM model or a base NLI model can reach an accuracy of roughly

Kown \ Unknown		SQ-ADV	TRIVIA	HOTPOT
SQ-ADV	MAXPROB	85.0	88.7	87.5
	KAMATH	88.8	89.5	88.9
	BOWPROP	91.5	90.6	89.0
	LIMECAL	**94.5**	**91.7**	**91.9**
TRIVIA	MAXPROB	85.0	88.7	87.6
	KAMATH	85.6	91.9	88.7
	BOWPROP	85.3	92.1	89.9
	LIMECAL	**90.9**	**92.5**	**92.1**
HOTPOT	MAXPROB	85.0	88.7	87.6
	KAMATH	86.1	91.4	89.4
	BOWPROP	85.1	91.8	91.6
	LIMECAL	**91.7**	**92.3**	**92.5**

Table 6: Area under Coverage-F1 curve in the Selective QA setting. Our explanation-based approach is also strong in this setting, substantially outperforming existing baseline and our own ablation.

80%. Our explanation-based approach largely lags glass-box methods, likely because the base NLI model utterly fails on QNLI (50.5% accuracy) and MRPC (55.0% accuracy) and does not grant much support for the two tasks. Nonetheless, the results on NLI still support our main hypothesis: explanations can be useful for calibration.

5 Selective QA Setting

Our results so far have shown that a calibrator can use explanations to help make binary judgments of correctness for a model running in a new domain. We now test our model on the selective QA setting from Kamath et al. (2020) (Figure 2). This experiment allows us to more directly compare with prior work and see performance in a setting where in-domain (ID) and out-of-domain (OOD) examples are mixed together.

Given a QA model trained on source domain data, the goal of selective QA is to train a calibrator on a mixture of ID source data and *known* OOD data, and test the calibrator to work well on a mixture of in-domain and an *unknown* OOD data.

We follow the similar experimental setup as in Kamath et al. (2020). The detailed setting is included in the supplementary material.

Results As shown in Table 6, similar to the main QA results. Our explanation-based approach, LIMECAL, is consistently the best among all settings. We point out our approach outperforms KAMATH especially in settings that involve SQUAD-ADV as known or unkown OOD distribution. This can be attributed the similarity between SQUAD and SQUAD-ADV which can not be well distinguished with features used in KAMATH (`Context Length`, `Answer Length`, and etc.). The strong performance of our explanation-based approach in the selective QA setting further verifies our assumption: explanation can be useful and effective for calibrating black box models.

6 Related Work

Our approach is inspired by recent work on the *simulation* test (Doshi-Velez and Kim, 2017), i.e., whether humans can simulate a model's prediction on an input example based on the explanations. Simulation tests has been carried out in various tasks (**?**Nguyen, 2018; Chandrasekaran et al., 2018; Hase and Bansal, 2020) and give positive results in some tasks (Hase and Bansal, 2020). Our approach tries to mimic the process that humans would use to judge a model's prediction by combining heuristics with attributions instead of having humans actually do the task.

Using "meta-features" to judge a model also appears in literature on system combination for tasks like machine translation (Bojar et al., 2017), question answering (Kamath et al., 2020; Zhang et al., 2021), constituency parsing (Charniak and Johnson, 2005; Fossum and Knight, 2009) and semantic parsing (Yin and Neubig, 2019). The work of Rajani and Mooney (2018) in VQA is most relevant to ours; they also use heuristic features, but we further conjoin heuristic with model attributions.

7 Discussion & Conclusion

Limitations Despite showing promising results in improving model generalization performance, our attribution-based approach does suffer from intensive computation cost. Using either LIME or SHAP to generate attributions requires running inference a fair number of perturbations when the input size is large (see Appendix for details), which limits our method's applicability. But this doesn't undermine the main contribution of this paper, answering the question in the title, and our approach is still applicable as-is in the scenarios where we pay for access to the model but not per query.

Conclusion We have explored whether model attributions can be useful for calibrating black box models. The answer is *yes*. By connecting attributions with light human heuristics, we successfully improve model generalization performance on new domains, or even different tasks. Besides, it exhibits promising generalization performance in some settings (cross-domain generalization and Selective QA).

References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*.

Arjun Chandrasekaran, Viraj Prabhu, Deshraj Yadav, Prithvijit Chattopadhyay, and Devi Parikh. 2018. Do explanations make VQA models more predictable to a human? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*.

Victoria Fossum and Kevin Knight. 2009. Combining constituent parsers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS)*.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Nazneen Fatema Rajani and Raymond Mooney. 2018. Stacking with auxiliary features for visual question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392, Austin, Texas. 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 ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*.

Lloyd S Shapley. 1997. A value for n-person games. *Classics in game theory*, 69.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xi Ye, Rohan Nair, and Greg Durrett. 2021. Connecting attributions and QA model behavior on realistic counterfactuals. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Shujian Zhang, Chengyue Gong, and Eunsol Choi. 2021. Knowing more about questions can help: Improving calibration in question answering. In *Findings of the Association for Computational Linguistics (ACL Findings)*.

A Details of the Kernel used in LIME and SHAP

LIME heuristically sets π_x as an exponential kernel (with bandwith σ) defined on the cosine distance function between the perturbation and original input, i.e.,

$$\pi_x(z) = \exp(-d_{\cos}(x, h_x(z))/\sigma^2)$$

That is, LIME assigns higher instance weights for perturbations that are closer to the original input, and so prioritizes classifying these correctly with the approximation.

SHAP derives the π_x so the ϕ can be interpreted as Shapley values (Shapley, 1997):

$$\pi_x(z) = \frac{n-1}{\binom{N}{|z|}|z|(n-|z|)}$$

where $|z|$ denotes the number of activated tokens (sum of z). This kernel assigns high weights to perturbations with few or many active tokens, as the predictions when a few tokens' effects are isolated are important. This distinguishes SHAP from LIME, since LIME will place very low weight on perturbations with few active tokens.

B Detailed Setup of Selective QA Setting

We follow the similar experimental setup as in Kamath et al. (2020). We train a ROBERTA QA model on SQUAD, and use on a mixture of 1,000 SQUAD dev examples + 1,000 known OOD examples to train the calibrator. We report test results on both the same type of mixture (1,000 SQUAD + 1,000 known OOD, diagonal blocks) and a mixture of 4000 SQUAD examples + 4,000 unkown OOD (2,560 SQUAD + 2,560 SQUAD-ADV as SQUAD-ADV only contains 2,560 examples).

C Feature Importance

We analyze the important features learned by the calibrator. We find explanation-based features are indeed generally among the top used features and more important than Bag-of-Word-based features (see the Appendix for a detailed list). All QA calibrators heavily rely on attribution values of the proper nouns (NNP) and wh-words in the question. BoW features of overlapping nouns are considered important on QNLI, but the top feature is still attribution-based.

These factors give insights into which parts of the QA or NLI reasoning processes are important for models to capture. E.g., the reliance on NNPs in SQUAD-ADV matches our intuitive understanding of this task: distractors typically have the wrong named entities in them, so if the model pays attention to NNPs on an example, it is more likely to be correct, and the calibrator can exploit this.

Table 7 shows the most important features learned by LIMECAL for QA and NLI. For brevity, we present the features related to the probabilities of the top predictions into one feature (`Prob`). Explanation-based features are indeed generally among the top used features and more important than raw property features.

D Details of POS Tag Properties

We use tagger implemented in spaCy API.[5] The tag set basically follows the Penn Treebank tag set except that we merge some related tags to reduce the number of features given the limited amount of training data.[6] Specifically, we merge `JJ,JJR,JJS` into `JJ`, `NN,NNS` into `NN`, `NNP,NNPS` into `NNP`, `RB,RBR,RBS` into `RB`, `VB,VBD,VBG,VBN,VBP,VBZ` into `VB`, and `WDT,WP,WP$,WRB` into `W`. In this way, we obtain a tag set of 25 tags in total.

E Details of Black Box Calibrators

Number of Feature for QA

- KAMATH (Kamath et al., 2020): we use the **7** features described in (Kamath et al., 2020), including `Probability` for the top 5 predictions, `Context Length`, and `Predicted Answer Length`.
- BOWPROP: In addition to the 7 features used in KAMATH. We construct the property space $\mathcal{V}$ as the union of low-level `Segment` and `Segment` $\times$ `Pos-Tags`. Since there are 3 segments `question`, `context`, `answer` in the input, and 25 tags (Section D), the size of the property space $|\mathcal{V}|$ is thereby given as $3 + 3 \times 25 = 78$. Therefore the total number of features (including the 7 from KAMATH) is **85**.
- LIMECAL and SHAPCAL: Recall that the size of the property space is 78. LIMECAL and SHAPCAL uses 78 features describing the attribution related to the corresponding properties in addition to the 85 features used in BOWPROP. The total number of features is therefore **163**.

[5] https://spacy.io/api

[6] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

SQ-ADV	TRIVIA	HOTPOT	QNLI	MRPC
Attr to NNP in Q	Prob of Top Pred	Prob of Top Pred	Attr Overlapping NN in H	Prob of Top Pred
Attr to VB in C	Answer Length	Attr to Q	BOW Overl- NN in H	Attr to P
Prob of Top Pred	Attr NNP in Q	Attr Wh- in Q	BOW Overl- NN in P	Attr to H
Attr to NN in Q	Attr Wh- in Q	Attr to C	Attr to Non-Overl- NN in P	Attr to Non-Overl- NNP in H
Answer Length	Attr to Question	Attr to NNP in Q	Prob of Top Pred	Attr to Overl- SYM in P

Table 7: Most important features used by the LIMECAL in different tasks. For QA, `Attribution of NNP in the question` and `Attribution of Wh- in the question` are generally important. For NLI, features related to overlapping/non-overlapping nouns are more effective.

Features Numbers for NLI

- CLSPROB (Kamath et al., 2020): we use **2** features in practice, `Probability of Entailment` and `Probability of Contradiction`. We do not include `Probability of Neutral` since it can be inferred from the probabilities of two other classes.
- BOWPROP: In addition to the 2 features used in CLSPROB, we construct the property space $\mathcal{V}$ as the union of low-level `Segment` and `Segment` $\times$ `Pos-Tags` $\times$ `Overalapping Words`. Since there are 2 segments (`Premise`, `Hypothesis`), 25 tags (Section D), and 2 properties for overlapping `Overlapping`, `Non-Overlapping`, the size of the property space $|\mathcal{V}|$ is given as $2 + 2 \times 25 \times 2 = 102$. Therefore the total number of features (including the 2 from CLSPROB) is **104**.
- LIMECAL and SHAPCAL: LIMECAL and SHAPCAL add another 102 features in addition to the 104 features used in BOWPROP. The total number of features are therefore **206**.

Costs for Generating Explanations For QA tasks which have relatively long inputs, we sample 2048 perturbations and run inference over them for each example. For simpler NLI tasks, we use about 512 model queries for each example.

Hyperparameters We use the RandomForest implementation from Scikit-Learn (Pedregosa et al., 2011). We list the hyper-parameters used in each approach in Table 8. The hyperparameters are determined through grid search using 400 training examples and 100 validation examples. The choices of numbers of trees are [200, 300, 400, 500], and choices of max depth are [4, 6, 8, 10, 15, 20]. Then, for the experimental results in Table 1, Table 2, and Table 3, we always fix the hyper-parameters, and do not perform any further hyper-parameter tuning.

QA		NUM. TREE	MAX DEPTH
SQ-ADV	KAMATH	300	6
	BOWPROP	300	20
	LIMECAL	300	20
	SHAPCAL	300	20
TRIVIA	KAMATH	300	6
	BOWPROP	300	10
	LIMECAL	300	20
	SHAPCAL	300	20
HOTPOT	KAMATH	300	4
	BOWPROP	300	10
	LIMECAL	300	10
	SHAPCAL	300	10
NLI		**NUM. TREE**	**MAX DEPTH**
QNLI	KAMATH	300	4
	BOWPROP	300	6
	LIMECAL	400	20
	SHAPCAL	400	20
MRPC	KAMATH	300	6
	BOWPROP	300	8
	LIMECAL	400	20
	SHAPCAL	400	20

Table 8: Hyper-parameters used to train RandomForest classifier for different approaches.

F Details of Glass Box Methods

Finetuning RoBERTa For QA, we finetune the ROBERTA-base model with a learning rate of 1e-5 for 20 epochs (We also try finetuning for 3 epochs, but the objective does not converge with 500 examples.) We set the batch size to be 32, and warm-up ratio to be 0.06.

For MNLI, we finetune a ROBERTA-base model with a learning rate of 1e-5 for 10 epochs. We set the batch size to be 32, and warm-up ratio to be 0.06, following the hyper-parameters in Liu et al. (2019).

Adapt Base QA/NLI Model For QA, we adapt the base ROBERTA QA model trained on SQUAD with a learning rate of 1e-5 for 2 epochs.

For MNLI, we finetune base ROBERTA NLI model trained on MNLI with a learning rate of 1e-

5 for 10 epochs. The objective does not converge when finetuning for 2 epochs, as the MNLI task is too different from QNLI and MRPC.