Investigating Ensemble Methods for Model Robustness Improvement of Text Classifiers

Jieyu Zhao\textsuperscript{1} Xuezhi Wang\textsuperscript{2} Yao Qin\textsuperscript{2} Jilin Chen\textsuperscript{2} Kai-Wei Chang\textsuperscript{1}

\textsuperscript{1}University of California, Los Angeles \textsuperscript{2}Google Research
jieyuzhao@ucla.edu {xuezhiw, yaoqin, jinlinc}@google.com kwchang@cs.ucla.edu

Abstract

Large pre-trained language models have shown remarkable performance over the past few years. These models, however, sometimes learn superficial features from the dataset and cannot generalize to the distributions that are dissimilar to the training scenario. There have been several approaches proposed to reduce model’s reliance on these bias features which can improve model robustness in the out-of-distribution setting. However, existing methods usually use a fixed low-capacity model to deal with various bias features, which ignore the learnability of those features. In this paper, we analyze a set of existing bias features and demonstrate there is no single model that works best for all the cases. We further show that by choosing an appropriate bias model, we can obtain a better robustness result than baselines with a more sophisticated model design.

1 Introduction

Advances in pre-trained language models have shown great performance in natural language processing (NLP) benchmarks. However, these models often learn dataset-specific patterns and cannot generalize well to out-of-distribution data (McCoy et al., 2019; Niven and Kao, 2019; Si et al., 2019; Ko et al., 2020). These patterns are referred to as bias features, which have strong indications of instance labels but do not necessarily generalize to out-of-distribution data (Geirhos et al., 2020). For example, in MNLI (Williams et al., 2018), the appearance of a negation word in an example has a strong correlation with label “contradiction” (Gururangan et al., 2018). A model leveraging such bias features can exhibit good performance on in-domain data but will break when evaluated on an out-of-distribution test set where the correlation between the patterns and labels no longer holds.

Given prior knowledge of possible bias features in the dataset, several approaches have been proposed to reduce models’ overreliance on the bias features (Clark et al., 2019; He et al., 2019; Utama et al., 2020). The underlying idea is to discourage the model to learn from “easy” examples that can be predicted correctly solely based on bias features. These works first train a bias model to capture bias patterns. They then train a main model and ensemble it with the bias model in a way that the predictions of the main model are adjusted by not leveraging the strategy captured by the bias model. Product-of-experts (Hinton, 2002) and self-distillation approaches have been widely adopted for the ensemble (Clark et al., 2019; He et al., 2019; Mahabadi and Henderson, 2019; Utama et al., 2020; Du et al., 2021).

Although these methods improve model robustness on some benchmark datasets, none of them study how to choose the bias model and only assume that a weak classifier (e.g., logistic regression) can explicitly capture the bias patterns. We argue that it is very important to choose the appropriate bias model. On one hand, different bias features may not be captured by one model with a fixed model size (capacity). For example, in MNLI, a model capturing the “negation word occurrence” does not necessarily capture the token overlap pattern at the same time. On the other hand, we do not expect an over-capacity bias model as it may capture non-bias features which will also be factored out during the ensemble, thus worsening the overall model performance. To do this, we train bias models with different capacities leveraging the product-of-experts method and compare with current state-of-the-art methods. We empirically show the different learnability of the bias patterns requires models with diverse capacities to better capture them. For example, a BERT-mini model can learn a token-overlap style bias pattern better in the MNLI dataset compared with the logistic...
regression used by existing literature.

In this work, we conduct a deep analysis of the existing literature on ensemble-based methods for model robustness improvement where most of them follow the hypothesis of adopting one simple model to learn bias patterns. Instead, we study different bias models and demonstrate their ability to capture the bias patterns varies. We propose an approach to selecting the “best” bias model by splitting the development set into “easy” and “challenge” subsets. By utilizing the best bias models for different bias features, we show that we can make the models more robust compared to existing baselines, on both natural language inference and fact-verification tasks.

2 Bias Model Selection

To understand to what extent our models can conquer the biases, for each bias, we create a bias training dataset only consisting of examples with this bias feature to reduce the impact of other biases. To evaluate model’s ability to overcome the bias, we create a corresponding challenge test set which is composed of examples with that specific bias feature and not following the distribution in the training set. Examples of such sets are in Sec. 3.3.

To obtain the best bias model for each bias feature, we first train various bias models with different capacities on the bias training set and ensemble them with the main model. We then evaluate the main model on the corresponding challenge set and choose the one exhibiting the best robustness result. Our model pipeline is shown in Figure 1.

In this work, following Clark et al. (2019), we leverage product-of-experts (PoE), a commonly used method for model robustness improvement, as an example to illustrate our argument. Given a dataset \( D = \{(x_i, y_i)\}_{i \in [1,n]} \), where \( y_i \in \{1,2,\ldots,C\} \), a bias model \( h(x_i; \theta_b) = \langle b_{i1}, b_{i2}, \ldots, b_{iC}\rangle \) and a main model \( f(x_i; \theta) = \langle p_{i1}, p_{i2}, \ldots, p_{iC}\rangle \) where \( b_{ij} \) and \( p_{ij} \) are the probabilities predicted for label category \( j \), the goal is to learn \( \theta \) that can make a correct prediction for an input example without using the patterns learned by the bias model. To achieve such a goal, PoE (Hinton et al., 2015) fuses the two models as

\[
\hat{p}_i = \text{softmax}(\log(p_i) + \log(b_i)).
\]

Figure 1: An overview of model pipeline. Bias model is trained on biased subset of the original training dataset (more details are in Sec. 3.3). Blue arrows stand for the gradient flow. We only use the main model when doing the evaluation.

During training, the model is optimized for the cross-entropy loss based on \( \hat{p} \). After training, only the main model \( f \) will be used for evaluation.

3 Experiment

In this section, we use two tasks (in English) to study the effects of different bias models. We analyze the best bias model for various bias features.

3.1 Dataset

Fact Verification. FEVER (Thorne et al., 2018) is a dataset for fact verification task. Each instance contains a claim sentence and an evidence sentence. The goal is to verify if the claim is Supported, Refuted or NotEnoughInfo with the evidence. We evaluate model robustness on Fever-Symmetric dataset (Schuster et al., 2019).

Natural Language Inference. The goal of the natural language inference (NLI) task is to identify the relationship (Entailment, Contradiction or Neutral) between the hypothesis and premise sentences. We use the MNLI dataset (Williams et al., 2018) for training and evaluate the model robustness on the HANS dataset (McCoy et al., 2019).

3.2 Bias Features

Schuster et al. (2019) show that for the FEVER dataset, only using the claim sentence can obtain comparable results to using both claim and evidence. Hence we use this CLAIM-ONLY feature as one of our bias features. Similarly, we add another type of bias feature, which only uses the evidence sentence to make the prediction, and we refer to this as EVIDENCE-ONLY bias.

Clark et al. (2019) list several bias features in MNLI dataset, such as whether all the tokens of the hypothesis appear in the premise (ALL-IN-P), whether the hypothesis is a subsequence in premise (H-IS-SUBSEQ), the percentage of words in hypothesis that are also in premise (PERCENT-IN-P), and
### 3.3 Bias and Challenge Sets

Our method leverages the known bias features and verifies model’s ability to overcome the biases based on the bias training and challenge test sets. To obtain the biased training set for MNLI, we follow Clark et al. (2019) to select the examples equipped for each bias features separately. For both ALL-IN-P and H-IS-SUBSEQ, they are strongly related to “entailment” label while NEG-IN-H is closely related to “contradiction” label. To build the CHALLENGE SET, we filter test examples with a corresponding bias (e.g., satisfying the ALL-IN-P) but the label does not follow the bias pattern in the training dataset (e.g., labels are not “entailment”).

### 3.4 Capacity for Bias Models

In this section, we verify the best bias model for each bias feature. We use a BERT-base model as the main model for both the FEVER and MNLI datasets. In terms of the capacity for the bias model, we consider different BERT models (from tiny to base) as well as the one used in the existing literature. For MNLI, a widely used bias model is a logistic regression model trained on some predefined biased features (Clark et al., 2019). For FEVER dataset, existing work (Utama et al., 2020) leverages a shallow non-linear classifier, in this work, we use a multilayer perceptron (MLP) model.

Table 1 shows the results on FEVER when we use the bias models with different capacities, from MLP to BERT-base. The first row “None” stands for a naive BERT-base model without any bias model. CLAIM-ONLY stands for the model’s performance on the challenge set we create. It suggests that to deal with the CLAIM-ONLY bias, using BERT-base as the bias model can better improve the robustness on the challenge set, at the same...
time, we see the model keeps its performance on the in-distribution dev set. Similarly, when dealing with the EVIDENCE-ONLY dev set, we see that using BERT-tiny as the bias model can improve the model robustness better than other choices without performance loss in the in-distribution data.

We compare our methods with two other baselines, one is Reweight, where a model is trained on the weighted dataset. The weight of example $x_i$ is $1 - b_{iy_i}$, where $b_{iy_i}$ is the probability of the bias model on the correct label $y_i$ (Clark et al., 2019). Another baseline is self-distillation (Utama et al., 2020), where the ensemble is based on the knowledge distillation (Hinton et al., 2015). It is a more complicated ensemble method than PoE and requires an additional teacher model. More details are provided in the Appendix.

We show that by fusing different bias logits together, we can provide a way to improve the model robustness. We compare our methods with existing literature which leverages the biases from a fixed model (Utama et al., 2020). In Table 3, PoEMLP refers to a baseline which uses the MLP as the bias model. In Table 4, we see that the best bias model dealing with one bias feature (e.g., ALL-IN-P) does not necessarily work best for another one (e.g., NEG-IN-H). By mixing the logits from the best bias models for each bias feature, a

<table>
<thead>
<tr>
<th>Bias Model</th>
<th>dev-m</th>
<th>dev-mm</th>
<th>ALL-IN-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>83.78±0.24</td>
<td>84.21±0.11</td>
<td>28.60±5.51</td>
</tr>
<tr>
<td>Logistic Reg.</td>
<td>83.52±0.13</td>
<td>83.80±0.12</td>
<td>38.87±1.77</td>
</tr>
<tr>
<td>BERTtiny</td>
<td>80.02±0.83</td>
<td>80.87±0.30</td>
<td>45.27±3.19</td>
</tr>
<tr>
<td>BERTmini</td>
<td>80.72±0.99</td>
<td>80.75±0.37</td>
<td>43.03±3.78</td>
</tr>
<tr>
<td>BERTsmall</td>
<td>81.98±0.42</td>
<td>82.27±0.14</td>
<td>40.67±1.70</td>
</tr>
<tr>
<td>BERTmedium</td>
<td>81.97±0.34</td>
<td>82.40±0.14</td>
<td>36.80±0.85</td>
</tr>
<tr>
<td>BERTbase</td>
<td>82.71±0.35</td>
<td>83.17±0.32</td>
<td>37.77±5.2</td>
</tr>
</tbody>
</table>

Table 4: Model accuracy on MNLI dev-matched, dev-mismatched and corresponding challenge set when dealing with ALL-IN-P bias. BERT-tiny is the best bias model for this bias feature. More results about other bias features are in appendix.

PoE method can outperform self-distillation which has a more sophisticated ensemble schema.

**Insights** We notice that the logits fused into the main models can play a significant role in the model performance. For example, we observe that an overconfident bias model can hurt the model performance in the in-distribution evaluation set. We also find a possible negative effect on the robustness result when dealing with the bias features not related to the test set. For example, in HANS, there are no instances with the NEG-IN-H bias, and fusing the logits from the bias model for NEG-IN-H sometimes hurts the performance on HANS (more in appendix). However, in most cases, we do not have access to the bias features in the test set, how to deal with the potential conflicts between different bias features remains an unexplored yet very important direction and we leave it for future study.

### 4 Related Work

Recent NLP models show great performance when evaluating on the in-distribution test set. However, such results might not hold when the test set is out-of-distribution. For example, Schuster et al. (2019) discover that a fact verification model may make a prediction by looking at the occurrence of certain phrases in the input example. Similar scenarios have been observed in other applications such as in visual question answering (Agrawal et al., 2018), and paraphrase identification (Zhang et al., 2019).

Several ensemble-based methods have shown improvement in model robustness when dealing with dataset biases (Clark et al., 2019; He et al., 2019; Mahabadi and Henderson, 2019; Utama et al., 2020). Such methods usually contain two components for the ensemble and focus on different ensemble strategies. In contrast, we analyze the components for the ensemble. Our work fills in the gap of a deeper understanding of the bias patterns in the dataset and provide a pipeline to choose the
best component for the ensemble so that we can improve model robustness.

5 Discussion

How to improve the model robustness has been an important research topic, and several approaches have been proposed for such a goal, ranging from dataset augmentation (Kaushik et al., 2019) to model architecture design (Lewis and Fan, 2018). Although several ensemble-based methods have shown great improvement, they treat all the dataset artifacts exactly the same way. In this work, we revisit such methods, and demonstrate that, not all the dataset artifacts are the same and they require different capacity models to deal with. Contrary to the common beliefs in the existing literature that a smaller-capacity model captures the bias features, our paper is the first that investigates the effect of the bias model size and shows that better robustness needs to be achieved by bias models with different capacities. We also show that by better leveraging the information learned from the dataset artifacts, a simple ensemble method can achieve a better or the same level of model robustness.

Limitations

Our study in this paper only considers some known bias features. While this setting is common in the literature, we argue that in real applications, it might be very hard to get such information. In addition, this work is based on the ensemble method for robustness improvement and the bias models obtained for PoE might not be the same for another method. There are other ways to achieve robustness such as adversarial training (Grand and Belinkov, 2019) which we leave for future study to show how they can be used to deal with various bias features.

Acknowledgements

We would like to thank all the reviewers for their valuable feedback.

References


A Appendix

Training details All the BERT model we used is from HuggingFace (Wolf et al., 2019). We use the default hyperparameters in Utama et al. (2020) to train the FEVER model, e.g., 3 epochs with a learning rate $5 \times 10^{-5}$. To train the FEVER model, we leverage the preprocessed training data as indicated in Schuster et al. (2019). For the MNLI dataset, we train the model using the default hyperparameters except that we train the model for 6 epochs to make all the models converge. All the results in the paper are the averaged value for 3 runs. We train our models with NVIDIA GeForce RTX 2080 Ti GPUs on Pytorch, and each epoch takes approximate 1 hour.

Self-Distillation Given a teacher model, which provides the prediction for input example $x_i$ as $\langle \hat{p}_{i1}, \ldots, \hat{p}_{iC} \rangle$, with the probability from the bias model $\langle b_{i1}, \ldots, b_{iC} \rangle$, the scaled teacher output for each example is computed as $s = \langle s_{i1}, \ldots, s_{iC} \rangle$, where

$$s_{ij} = \frac{1 - b_{ij}}{\sum_{k=1}^{C} (1 - b_{ik})} \hat{p}_{ij}.$$

Then a model is trained to minimize the cross entropy loss between the scaled teacher output and the current output of the main model.

Bias model results We list the results on the dev and corresponding CHALLENGE sets for the MNLI task when training on the H-IS-SUBSEQ (in Table 6) and NEG-IN-H (in Table 7) bias features separately.

<table>
<thead>
<tr>
<th>Bias Model</th>
<th>dev-m</th>
<th>dev-mm</th>
<th>H-IS-SUBSEQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>86.10±0.13</td>
<td>88.10±0.79</td>
<td>14.40±4.44</td>
</tr>
<tr>
<td>Logistic Reg.</td>
<td>92.47±0.08</td>
<td>94.18±1.21</td>
<td>26.47±2.86</td>
</tr>
<tr>
<td>BERTtiny</td>
<td>81.45±0.27</td>
<td>81.75±0.50</td>
<td>29.53±3.87</td>
</tr>
<tr>
<td>BERTmini</td>
<td>79.51±0.10</td>
<td>80.35±0.11</td>
<td><strong>33.77±4.17</strong></td>
</tr>
<tr>
<td>BERTsmall</td>
<td>80.29±0.10</td>
<td>80.84±0.11</td>
<td>30.70±7.64</td>
</tr>
<tr>
<td>BERTmedium</td>
<td>80.39±0.24</td>
<td>80.94±0.24</td>
<td>27.13±1.35</td>
</tr>
<tr>
<td>BERTbase</td>
<td>81.47±0.27</td>
<td>81.69±0.39</td>
<td>29.93±5.20</td>
</tr>
</tbody>
</table>

Table 6: Results on dev and challenge sets for MNLI dataset when dealing with the H-IS-SUBSEQ bias feature.

Bias features vs. Model robustness We show here that dealing with the NEG-IN-H bias can potentially have negative effect on the test result. For example, when we fuse the logits from a BERT-tiny model only on the examples having the NEG-IN-H bias feature, we observe the accuracy on
<table>
<thead>
<tr>
<th>Bias Model</th>
<th>dev-m</th>
<th>dev-mm</th>
<th>NEG-IN-H</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>86.10±0.13</td>
<td>88.10±0.79</td>
<td>77.15±0.47</td>
</tr>
<tr>
<td>Logistic Reg.</td>
<td>92.47±0.08</td>
<td>94.18±1.21</td>
<td>77.69±1.01</td>
</tr>
<tr>
<td>BERTtiny</td>
<td>82.87±0.21</td>
<td>83.29±0.05</td>
<td>77.06±0.77</td>
</tr>
<tr>
<td>BERTmini</td>
<td>82.63±0.15</td>
<td>82.91±0.25</td>
<td>75.18±1.10</td>
</tr>
<tr>
<td>BERTsmall</td>
<td>82.00±0.10</td>
<td>82.13±0.06</td>
<td>73.29±0.16</td>
</tr>
<tr>
<td>BERTmedium</td>
<td>80.33±0.37</td>
<td>80.57±0.27</td>
<td>72.37±1.40</td>
</tr>
<tr>
<td>BERTbase</td>
<td>83.55±0.12</td>
<td>84.06±0.23</td>
<td>60.84±2.89</td>
</tr>
</tbody>
</table>

Table 7: Results on dev and challenge sets for MNLI dataset when dealing with the NEG-IN-H bias feature.

HANS drops from 63.05% to 59.93%, while fusing ALL-IN-P or H-IS-SUBSEQ achieves the accuracy 65.39% and 63.65% respectively.