Using Calibrator to Improve Robustness in Machine Reading Comprehension Without Performance Sacrificing

Anonymous ACL submission

Abstract

Machine Reading Comprehension(MRC) has achieved a remarkable result since some powerful models, such as BERT, are proposed. However, these models are not robust enough and vulnerable to adversarial input perturbation and generalization examples. Some works tried to improve the performance on adversarial perturbation by adding related examples into training data while it leads to degradation on the in-domain dataset, because the shift of data distribution makes the answer ranking based on the softmax probability of model unreliable. In this paper, we propose a method to improve the robustness by using a calibrator as the post-hoc reranker, which is implemented based on XGBoost model. The calibrator combines both manual features and representation learning features to rerank candidate results. Experimental results on adversarial datasets show that our model can achieve performance improvement by more than 10% and also make improvement on the in-domain and generalization datasets.

1 Introduction

Assisted by large pre-trained models, Machine Reading Comprehension(MRC) has achieved human-comparable results on some existing datasets. But even state-of-the-art (SOTA) models trained on such datasets are not robust enough. These models are not only vulnerable to adversarial input perturbations, but also perform poorly on out-of-domain data.

Building more challenging MRC datasets may improve the robustness, but the whole process is expensive. Therefore, there are two directions to address the problem based on existing datasets. One is the data level. Using some of adversarial or out-of-domain examples as data augmentation can improve performance on corresponding dataset, but it leads to degradation on the in-domain dataset. The other is the model level. Adding complex structures in models and modifying loss function may improve generalization and defend adversarial attack, but the new model is time-consuming and memory intensive during training and inference.

In this paper, we proposed a simple yet effective method to improve performance on adversarial and generalization datasets without sacrificing in-domain performance in extractive MRC task. We applied several kinds of adversarial examples to explore the vulnerability of SOTA MRC model, and we found that the reason for the performance degradation was not that the model completely lost its ability to predict the range of correct answers, but that the ranking of candidate answers became unreliable. In other words, the model can still predict the correct range, but won't choose it as final output. Based on the above observation and inspired by previous work, we proposed a method, in which a calibrator is used as the post-hoc reranker to adjust the ranking of candidates. On account of the time complexity and space consumption, we adopted XGBoost to implement the calibrator.

Instead of BERT (Devlin et al., 2019), we use RoBERTa (Liu et al., 2019) as our backbone MRC model, for the latter shows higher level of robustness in MRC task. We use SQuAD 2.0 dataset (Rajpurkar et al., 2018) as main dataset and use Natural Questions (Kwiatkowski et al., 2019) to reveal generalization ability. We employ the methods proposed by Maharana and Bansal (2020) to generate adversarial examples, which is diverse and has been proved aggressive to attack baseline MRC models. And then we utilize our proposed calibrator as a post-hoc reranker to improve robustness.

Our contributions can be summarized as follows:

- We had a thorough research on adversarial examples generation on MRC datasets and made an analysis with statistical data of the influence of these examples on MRC models.
- We proposed a simple yet effective method to use calibrator as a reranker to improve perfor-

mance on adversarial datasets without sacrificing in-domain performance.

- We expand the feature space of calibrator by introducing two new manual features and integrating representation learning features to characterize model's states during inference, while previous works are limited to focusing on shallow manual features only.

2 Related Work

Robustness in MRC Robustness is a research highlight in NLP because researchers have found that models achieved impressive performance on particular datasets is too vulnerable for practical application (Jin et al., 2020). As for MRC, the research on robustness of models can be generally categorised into two directions: generalization to out-of-domain distributions and robustness under test-time perturbations (Si et al., 2021a). Both directions will disturb the data distribution, but they have different goals. Adversarial input perturbations aim to ascertain whether model learns shortcut, which means model learns to answer questions based on some specific implicit patterns rather than reading comprehension ability (Lai et al., 2021). Generalization aims to extend application scope of the model to out-of-domain data and maintains performance under domain-shift (Kamath et al., 2020). Many previous researches focus on exposing models' vulnerabilities through maliciously designed inputs and bringing forward to new challenging datasets and tools (Gan and Ng, 2019; Sen and Saffari, 2020; Jin et al., 2020; Si et al., 2021a; Bartolo et al., 2021; Si et al., 2021b). Another perspective is to modify the model structure and loss function, such as introducing external knowledge and multi-task strategy (Wu and Xu, 2020), adding adapters (Han et al., 2021), changing loss function to adjust bias caused by generalization (Wu et al., 2020; Liu et al., 2020) and so on. These models are more robust but have more than doubled parameters.

Adversarial Examples Generation The goal of adversarial attack is to mislead the model into giving wrong outputs. Due to discrete characteristics of Natural Language, some aggressive adversarial attack methods in Computer Vision may cause out-of-distribution(OOD) problem in NLP. As for MRC, adversarial input perturbation on contexts and questions may have a great effect. There are various ways to perturb the text of contexts and questions, such as word substitution, heuristics, gradient-based techniques and so on (Zhang et al., 2019; Bao et al., 2021). Jia and Liang (2017) first proposed to use distracting sentences that have significant overlap with the question and insert them into the context to generate adversarial examples. However, the creation of such distracting sentences is based on fixed templates, so the model probably identifies learnable biases and overfits to the templates instead of being robust to attack itself (Maharana and Bansal, 2020). Then more researches have tried to address this problem by creating complex templates (Wang and Bansal, 2018), or exploring more challenging generative methods (Gan and Ng, 2019; Si et al., 2021b; Bartolo et al., 2021). In addition to adding confusing sentences into contexts, there are several methods that can be aggressive and cause huge performance degradation as well, such as deleting pivotal sentences from contexts (Maharana and Bansal, 2020), using language models to generate new questions with same semantics and different syntactic forms (Iyyer et al., 2018), perturbing word embedding (Lee et al., 2021) and so on. Maharana and Bansal (2020)'s work is comprehensive by containing inserting distracting sentences, deleting crucial sentences and paraphrasing questions, so we apply their method to generate adversarial examples and analyze their influence.

Calibration in NLP The question of whether the model's confidence provides an accurate empirical measure of how likely the model is to be correct has been put forward to examine the reliability of the model (Jung et al., 2020; Jiang et al., 2021). A well-calibrated model should ensure that the confidence of its predictions is consistent with its accuracy, which means it shouldn't output incorrect predictions with high confidence. Previous works have found that the model which gives good confidence estimates on in-domain data is overconfident on OOD data (Desai and Durrett, 2020). In MRC, models tend to choose results with maximal softmax probability as final outputs. But out-of-domain data leads to the shift of data distribution, which causes overconfident issue (Kamath et al., 2020; Xin et al., 2021). Previous works proposed to apply the calibrator as a threshold to decide whether to abstain the prediction and try to avoid making confident yet incorrect predictions in preserved examples (Kamath et al., 2020; Xin et al., 2021; Zhang et al., 2021). Based on the analysis of the impact of adversarial examples, instead of using it as a threshold, we use the calibrator as a reranker.

3 Method

We use the calibrator as a post-hoc reranker to improve robustness in extractive MRC task. Basic QA model feeds outputs and some important model features into the calibrator and then calibrator chooses the best answer span from k[1] candidates as final outputs. We follow prior works (Kamath et al., 2020; Zhang et al., 2021), for the idea and basic features. But the key differences are that we adopt different calibrator architecture and use it for reranking rather than as a discard threshold ,and we extend the feature space. We categorized features into two kinds: manual features that are irrelevant to the MRC model, and representation learning features that revealed model states.

3.1 Metrics

Previous works (Kamath et al., 2020; Zhang et al., 2021) use the calibrator to decide whether to abstain an example, so the metrics to evaluate calibrator performance are associated with accuracy of binary classification and performance of the retained examples. They first plot risk versus coverage graph, where coverage is the fraction of evaluation data that calibration chooses to retain and risk is the error at that coverage. And they calculate the area under the curve, i.e. AUROC(Area Under the Receiver Operating Characteristics Curve), as the metrics.A good calibrator should cover as much coverage as possible with a specific given accuracy.

We propose to use the calibrator to choose the best from candidates, so it is a multi-classification problem rather than binary classification as previous work. And we don't abstain examples, so we use a different metric to evaluate the performance of calibrator, which is classification accuracy.

To measure MRC task performance, we use the answers chosen by calibrator as final outputs, and measure F1 score as a metric like common extractive MRC task.

3.2 Basic MRC model

We choose RoBERTa-large (Liu et al., 2019) as our backbone model for its superior performance and relative robustness. And we use standard span prediction architecture for extractive MRC task.

We remain the architecture of MRC model. The model has same input format and training process as general MRC models. But we make minor changes to its final outputs. After training, in addition to outputting unique id and text of answer with maximal softmax probability for each example as usual, the model also needs to output some features generated during inference, which will be described in section 3.4 and 3.5.

[1] k is set to 10 in our experiments.

3.3 Calibrator architecture

We apply gradient boosting library XGBoost (Chen and Guestrin, 2016) to train a multi-classifier to chooses one answer from k candidates provided by the baseline MRC model. The calibrator does not share its weights with basic MRC models. Since our target is to prove the effect of calibrator on adversarial datasets, we simply keep most of hyperparameters as their default values: max depth, subsample, colsample by tree and so on. To accelerate the training and inference process, we set the number of estimators to 160 and set the learning rate to 0.1. There may be some space for improvement by tuning these hyperparameters, but we focus on the overall effect of calibrator on adversarial examples, so there is no experiment related to tuning hyperparameters.

3.4 Manual features

As said before, manual features are completely irrelevant to the model, but characterize the property of data.

We use the following features for each input example i: q_i and c_i indicate the text length of corresponding question and context respectively, K_i is the collection of its k candidates. For each candidate k_{ij} in K_i where j is the original ranking in the candidates, we denote its features with a quadruple: $k_{ij} = (l_{ij}, p_{ij}, s_{ij}, e_{ij})$, where l_{ij} means the text length, p_{ij} indicates corresponding softmax probability, s_{ij} and e_{ij} refer to start logits and end logits respectively.

Inspired by previous work, we proposed two heuristic features based on a small amount of additional calculation on the above features.

One is to calculate the entropy according to general formula based on the softmax probability of top k predictions as the entropy feature E_i:

$$E_i = -\left[\sum_{j=1}^{k} p_{ij}\log p_{ij} + \left(1-\sum_{j=1}^{k} p_{ij}\right)\log\left(1-\sum_{j=1}^{k} p_{ij}\right)\right] \quad (1)$$

The reason we use entropy instead of other transformations is that the entropy of distribution over candidates can inform the calibrator of how uncertain the model is with respect to the question. This statement has been demonstrated on other question answering tasks using generative models in Jiang et al. (2021), so we assume it is effective in our model and test it in our robustness experiments.

The other is based on the calculation of softmax probability. When calculating the softmax probability for each candidate prediction, start and end logits are added as final score. The MRC model then use these softmax probabilities as confidence to choose the final answer. But the shift of data distribution leads to overconfident problem. Inspired by Guo et al. (2017), we use a single scaling factor T to alleviate the problem. Temperature scaling can soften the softmax with T>1. The whole calculation is as follows:

$$score_{ij} = \frac{s_{ij} + e_{ij}}{T} \tag{2}$$

$$sp_{ij} = \frac{e^{score_{ij}}}{\sum_{j=1}^{k} e^{score_{ij}}} \tag{3}$$

When the temperature scaling factor T is set to 1, sp_{ij} is equal to p_{ij} (sp means "softed probability"). To address overconfident issue, we set T to 1.3, which is acquired through several experiments.

So we take manual features with a total of $3+5k$ into consideration.

3.5 Representation learning features

The other category is based on specific representations from models. When the batch size is set to 1 during inference process, the states of trained model is relevant to the input example and may imply information about selecting optimal answer.

For each input example i containing a question and a context, the pipeline will separate them with a special token, and generate the embedding and a sequence of hidden vectors from different hidden layers. The prediction is generated based on the final hidden layer. We denote the embedding as v_i, which is a fixed dimensional vector. And we denote the hidden states of model as a sequence of vectors $h_i = (h_{i,0}, h_{i,1}, ..., h_{i,n})$, where n is the number of layers [2] and $h_{i,m}$ is the corresponding hidden vector of m-th hidden layer. The vectors in $h_{i,m}$ have the same dimensionality as the embedding vector v_i, and we denote the dimensionality as l.

[2]For RoBERTa-large, n is 24

The large scale of h_i may induce slow training and inference. So we only consider the vector $h_{i,n}$ from last hidden layer and the average vector A_i calculated as follows:

$$A_i = \frac{1}{n} \sum_{m=1}^{n} h_{i,m} \tag{4}$$

And we discovered that adding embedding output v_i is more effective, so we modify the calculation of A_i to:

$$A_i = \frac{1}{n+1} \left(\sum_{m=1}^{n} h_{i,m} + v_i \right) \tag{5}$$

As a conclusion, we get three vectors v_i, $h_{i,n}$ and A_i from the extractive MRC model. The three vectors have same dimensionality l, so we take representation learning features with a total of $3l$ into consideration.

4 Experiments

4.1 Experiments settings

We take RoBERTa-large (Liu et al., 2019) provided by Hugging face transformers as our basic MRC model and use XGBoost (Chen and Guestrin, 2016) provided by python library as the calibrator.

We choose SQuAD 2.0 dataset (Rajpurkar et al., 2018) as our main dataset, and first fine-tune basic RoBERTa-large model on the training dataset with two epochs. Then we randomly extract 10k samples from the training set and the validation set of SQuAD 2.0 respectively, and use the methods of adversarial examples generation provided in Maharana and Bansal (2020) to generate adversarial examples on these data. We use adversarial data generated on samples from validation set as our test set for robustness studies. The adversarial data generated on samples from training set is used to train the calibrator. And we also separate half of SQuAD 2.0 validation set for calibrator training and use the rest for evaluation.

We use Natural Questions dataset (Kwiatkowski et al., 2019) to evaluate generalization performance, because Natural Questions is generated from Wikipedia like SQuAD dataset (Rajpurkar et al., 2018) but with wider coverage. For convenience, we follow the setting of Sen and Saffari (2020) and use the provided scripts to convert Natural Questions datasets into a shared SQuAD 2.0 JSON format. We also use the same metrics for better comparison with original SQuAD 2.0 dataset. See Appendix A for some data examples.

4.2 Adversarial attack and generalization

Followed Maharana and Bansal (2020), the methods of adversarial examples generation can be divided into two categories according to whether the language model is used in the process: negative for those are independent of language models and positive for the opposite.

The negative category contains four methods: AddSentDiverse, AddKSentDiverse, AddAnswerPosition, and InvalidateAnswer. These methods use templates or heuristics to generate distracting sentences and then insert them randomly into context, or apply deletion of crucial sentences to disturb the model. The positive category is composed of two methods: PerturbAnswer and PerturbQuestion. Both methods use language model to rephrase sentences into different forms with the same semantics. The detailed description and examples of these methods can refer to Maharana and Bansal (2020). Considering that AddKSentDiverse has the same principle as AddSentDiverse but is more aggressive, we only use AddKSentDiverse. PerturbAnswer is not suitable for our experimental scenario either, because our main dataset is SQuAD 2.0 that contains unanswerable questions. In summary, we apply four kinds of methods to generate adversarial examples: AddKSentDiverse, AddAnswerPosition, InvalidateAnswer, and PerturbQuestion.

We use adversarial examples generated on samples from validation set as parts of test sets, and those generated on samples from training set to train the calibrator. Table 1 shows sizes of each test set and the results of evaluating basic model, where the model trained on SQuAD 2.0 merely chooses the answer with maximal softmax probability as output without using calibrator. The results show that adversarial examples are aggressive to basic MRC model, among which PerturbQuestion is the most aggressive, resulting in the most decline(from 87.39 to 45.27). Due to the impact of data amount, the size of adversarial examples used to train the calibrator is 5k each. According to Maharana and Bansal (2020), adding adversarial examples to train the basic model makes great improvement on adversarial datasets while degradation on in-domain dataset. And our experiments confirmed it by adding adversarial examples 5k each kind into training data of model and showing results in AD column of table 1. The in-domain performance drop from 87.39 to 85.88 while generalization performance drop from 53.30 to 51.92. The impact of adversarial examples on model trained on in-domain data only is described on section 5.1.

Testset	Size	F1(base)	F1(AD)
SQuAD2.0-dev	5937	87.39	85.88
AddKSentDiverse	4586	53.41	81.81
AddAnswerPosition	4355	68.72	85.08
InvalidateAnswer	5861	65.96	93.82
PerturbQuestion	3923	45.27	64.42
Natural Questions	3369	53.30	51.92

Table 1: Data scale and results without using calibrator on six test datasets. Base column represents results of baseline model after training on SQuAD 2.0 dataset. AD means adversarial training and this column represents results of baseline model after training on the mixture of in-domain and adversarial data.

4.3 Calibrator

A good calibrator should improve the performance on adversarial and generalization dataset, and maintain even improve the performance on the in-domain dataset. We use data described in section 4.1 to train and evaluate the calibrator.

We hypothesis that if qualified features are extracted, the calibrator can improve performance on the distribution-shift datasets even trained on in-domain data. But since the calibrator is ignorant of the type of distribution-shift data, it can't utilize representation learning features and just maintain the baseline result. Manual features can be helpful but its role is limited. So we conclude that calibrator is effective with the help of distribution-shift data, while it maintains the baseline when trained only on in-domain data. See Appendix B for result and more details.

Therefore, in our main experiments, the calibrator is trained in two settings: Single Mixed and All Mixed. Single Mixed means the calibrator is only trained on the mixture of one kind of adversarial data and hold-out in-domain data, and evaluate on corresponding test set, in-domain test set and generalization test set. All mixed data means the calibrator is trained on the mixture of all kinds of adversarial data and hold-out in-domain data, and evaluate on all test sets.

4.3.1 Single mixed data

As said in section 4.1, we train the calibrator on the mixture of in-domain data and 5k training adversarial examples, and evaluate on corresponding adversarial test set, in-domain and generalization

Trained on AddKSentDiverse+SQuAD		AddKSentDiverse		SQuAD 2.0 dev		Natural Questions	
Feature kind	Feature selection	Acc	F1	Acc	F1	Acc	F1
Baseline(without calibrator)		55.04	53.41	86.39	87.39	59.75	53.30
Manual	$c_i + q_i + l_{i0}$	56.56	55.55	85.43	86.69	56.93	52.71
	$+p_{ij}$	64.33	64.46	83.21	84.51	59.31	54.46
	$+p_{ij} + E_i$	64.26	64.54	83.29	84.48	59.51	54.53
	$+sp_{ij}$	64.48	64.46	83.43	84.74	59.78	54.45
	$+sp_{ij} + E_i$	64.74	64.74	83.32	84.64	**60.14**	**54.58**
Representation learning	$+v_i$	56.56	55.43	85.33	86.58	58.68	52.87
	$+h_{i,n}$	65.94	66.99	85.60	86.59	59.84	53.62
	$+A_i$	64.83	65.99	86.14	87.26	59.72	53.41
Manual+ Representation learning	$+v_i + sp_{ij} + l_i$	62.87	62.55	85.14	86.28	59.31	53.43
	$+h_{i,n} + sp_{ij} + l_i$	67.03	67.84	85.43	87.16	59.54	53.92
	$+A_i + sp_{ij} + l_i$	**67.14**	**68.24**	86.29	**87.41**	59.75	53.38

Table 2: The results on AddKSentDiverse when calibrator is trained on the mixture of hold-out in-domain data and 5k AddKSentDiverse data. The description of features is in section 3 and details about test data are in section 4.1.

test set. We take manual features and representation learning features described in section 3 into consideration. Accuracy of calibrator and F1-score are the metrics to be evaluated. We take AddKSentDiverse as a representative to demonstrate varying results under different selection of features in table 2. The results of baseline are the same as corresponding results in table 1.

Table 2 shows that the access to target examples can bring great improvement on target test set. When only exploring manual features, the performance on the target testset can be improved by 11% on all metrics while degradation on in-domain dataset by about 3%. Among manual features, E_i and sp_{ij} we proposed can be most effective in improving performance, especially generalization performance. Under the feature combination of c_i,q_i,l_i0, $E_i$and sp_{ij}, the calibrator can improve the adversarial performance from 53.41(baseline) to 64.74, and improve generalization performance from 53.30 to 54.58 on F1 score, while degradation on in-domain dataset by less than 3%.

Representation learning features can be great helpful not only to improve the target performance by 13% but also to keep in-domain performance drop less than 1% on F1 score. The combination of manual features and representation learning features can improve the target performance by nearly 15% on F1 score, and improve in-domain performance. Under the best feature combination of c_i,q_i,l_i0,A_i,sp_{ij} and l_i, the calibrator can improve the adversarial performance from 53.41 to 68.24 on F1 score while maintain and even slightly improve the performance on in-domain and generalization dataset. This suggests that representation learning features can be informative for calibrator to adjust ranking problem caused by adversarial examples.

Due to the limitation of paper length, we can't list results of all feature combinations on all adversarial test sets, which will be available in our repository.

4.3.2 All mixed data

Under this setting, we train the calibrator on the mixture of 5k each of all kinds of adversarial data and hold-out in-domain data, and evaluate on all test sets. For a clear representation, we only list the results under the best feature combination of c_i,q_i,l_i0,A_i,sp_{ij} and l_i for comparison with single mixed setting in table 3. To be more specific, the results on Single Mixed column are obtained through four experiments under best feature selection on four adversarial examples respectively, each with the same setting as described in section 4.3.1. The results of in-domain and generalization test set on Single Mixed column are the average of four experiments. And the results on All Mixed column are obtained through one experiment, where the calibrator is trained on the mixture of all adversarial and hold-out in-domain examples under best feature selection. The result of all test sets under different feature selections will be available in our repository as well.

Test set	Single Mixed Acc	Single Mixed F1	All Mixed Acc	All Mixed F1
AddKSentDiverse	55.04+12.10	53.41+14.83	55.04+9.85	53.41+12.24
AddAnswerPosition	64.73+14.49	68.72+15.09	64.73+7.16	68.72+7.38
InvalidateAnswer	81.78+13.41	65.96+13.62	81.78+3.73	65.96+3.87
PerturbQuestion	29.72+12.08	45.27+7.96	29.72+12.26	45.27+8.40
Natural Questions	59.75+0.30	53.30+0.52	59.75+2.60	53.30+1.29
SQuAD 2.0 dev	86.39+0.10	87.39+0.05	86.39-0.08	87.39-0.03

Table 3: The best results on all datasets. The numbers before '+' are the baseline result represented in table 1, and the numbers after '+' are the improvements after using calibrator to reselect final results. The meaning of Single Mixed, All Mixed, best feature selection and more details are described in section 4.3.2.

Testset	Size	Better	Prop(%)
SQuAD2.0-dev	11873	1530	12.89
AddKSentDiverse	4586	2062	44.96
AddAnswerPosition	4355	1536	35.27
InvalidateAnswer	5861	1068	18.22
PerturbQuestion	3923	2757	70.28
Natural Questions	3369	1356	40.25

Table 4: The result on the number of examples with better candidates among top k candidates on all datasets.

The improvement on adversarial test sets under this setting is not as good as Single Mixed except PerturbQuestion. The reason may be the data distribution becomes more diverse with the incorporation of multiple types of adversarial examples. This diversity makes it harder for the calibrator to defend against adversarial attacks, but helps improve generalization ability. As for PerturbQuestion, the test set consists of adversarial examples generated through rephrasing questions. So the reason may be that model has better ability to understand rephrased sentences under All Mixed.

The results show that the effect of the calibrator is not limited to particular dataset. Our calibrator can improve performance on adversarial and generalization test sets without in-domain performance sacrificing whether trained on single or all mixed data. Previous work using adversarial examples as data augmentation to train the basic MRC model will lead to degradation on in-domain performance, as we give in table 1. We propose to use adversarial examples to train the calibrator instead, and with the help of manual features and representation learning features, this method can improve robustness while maintaining in-domain performance.

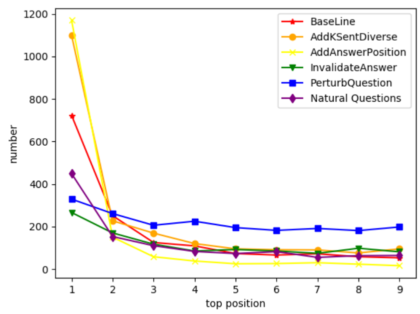

Figure 1: The label of best answers among top k candidates. We must emphasis that top 0 means the answer with max softmax probability instead of top 1.

5 Analysis

5.1 Analysis of baseline bad cases

In order to figure out why the performance of fine-tuned model dropped dramatically when applying adversarial or generalization examples, we analyzed the bad cases based on results in table 1. We defined any example whose final prediction has lower F1-score than average as a bad case. Then we explored the top k candidates provided by the model corresponding to this bad case, calculated the F1-score separately, and marked the best of top k candidates. If the answer with max softmax probability is not the best, it means there are better candidates in top$k$ predictions. We first made statistics on the number of bad cases in all datasets and proportion of examples with better candidates. We found that almost 90% of bad cases can find a better candidate among top k predictions. We also make this analysis on all examples of the whole dataset rather than limited to bad cases. We found that larger proportion of examples with better candidates in adversarial and generalization dataset

	In-domain	AddSent	AddOneSent
R.M-Reader(Hu et al., 2018)	86.6	58.5	67.0
KAR(Wang and Jiang, 2019)	83.5	60.1	72.3
BERT+Adv(Yang et al., 2019)	**92.4**	63.5	72.5
Sub-part Alignment(Chen and Durrett, 2021)	84.7	65.8	72.7
Our BERT-base	88.6	64.8	72.8
+ calibrator	88.5	**67.1**	**76.4**

Table 5: Performance of our method compared to previous robust MRC model on both AddSent and AddOneSent. The results are F1 scores on the full test set. The results show that we don't trade in-distribution performance to improve the model's robustness like previous work. More details are described in section 5.3.

comparing to only less than 13% of in-domain dataset. The results are represented in table 4. And the more aggressive the adversarial examples are, the higher the proportion of examples with better candidates(70% for PerturbQuestion).

So we came to the conclusion that the shift of data distribution makes the ranking based on softmax probability of baseline model unreliable. We used the labels of best among top k candidates to draw a line chart to show the shift in alignment between examples of high confidence and empirical likelihoods, which is presented in figure 1. Take AddKSentDiverse dataset as an example. There are more than 1k samples of this dataset with the best result ranked at position 1 (which is the second on the original ranking) instead of the top one with max softmax probability. From the graph, we found that most of best answers are limited to top 3 answers, which means the shift of data distribution didn't cause huge deviation on the ranking. So the calibrator used to rerank the candidates can make great improvement on adversarial datasets and improve the robustness.

5.2 Analysis of features selection

The selection of features is crucial to the effect of calibrator no matter which dataset. From section 4.3, manual features are informative to improve generalization performance while representation learning features perform better on in-domain and adversarial datasets. sp_{ij}, E_i and A_i can be helpful for various adversarial examples, and l_ij is most useful for InvalidateAnswer dataset due to the special way this dataset is constructed.

When multiple features are selected, the order of different features will have a certain impact on the results, but the impact is not as much as the selection of features. So results we reported are based on a random selection of permutations. More kinds of features and their combinations need further exploration.

5.3 Comparison to previous work

In Table 5, we compare our model under best feature selection with previous adversarial QA models in the literature. To make a fair comparison, we use BERT-base (Devlin et al., 2019) as our backbone model and use SQuAD 1.1 dataset (Rajpurkar et al., 2016) as our main dataset like previous work. We use 10k training examples of SQuAD 1.1 dataset to generate AddSentDiverse examples. We don't save in-domain examples and only use adversarial examples to train the calibrator. The method of Yang et al. (2019) works well on in-domain test set due to huge data augmentation. Besides, our method can guarantee the best in-domain performance.

6 Conclusion

We demonstrate that the impact of distribution-shift data on model is to make final ranking unreliable. So we use the calibrator as a reranker to improve performance of adversarial and generalization dataset without sacrificing in-domain performance. We take manual features and representation learning features into consideration while previous work only focus on manual features. When the calibrator is trained on the mixture of in-domain and adversarial data, the target performance can improve by more than 10% and generalization performance can improved by 1% while maintaining in-domain performance. And our calibrator only takes about ten minutes to train and is very easy to use as a post-hoc structure behind MRC model. To summarize, our calibrator is simple, effective, and has potential to be practical application and extended to other NLP tasks.

References

Rongzhou Bao, Jiayi Wang, and Hai Zhao. 2021. Defending pre-trained language models from adversarial word substitution without performance sacrifice. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3248–3258, Online. Association for Computational Linguistics.

Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jifan Chen and Greg Durrett. 2021. Robust question answering through sub-part alignment. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1251–1263, Online. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

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)*, pages 295–302, Online. Association for Computational Linguistics.

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.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861, Online. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Taehee Jung, Dongyeop Kang, Hua Cheng, Lucas Mentch, and Thomas Schaaf. 2020. Posterior calibrated training on sentence classification tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2723–2730.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yuxuan Lai, Chen Zhang, Yansong Feng, Quzhe Huang, and Dongyan Zhao. 2021. Why machine reading comprehension models learn shortcuts? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 989–1002, Online. Association for Computational Linguistics.

Seanie Lee, Minki Kang, Juho Lee, and Sung Ju Hwang. 2021. Learning to perturb word embeddings for out-of-distribution QA. 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 5583–5595, Online. Association for Computational Linguistics.

Kai Liu, Xin Liu, An Yang, Jing Liu, Jinsong Su, Sujian Li, and Qiaoqiao She. 2020. A robust adversarial training approach to machine reading comprehension. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8392–8400.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Adyasha Maharana and Mohit Bansal. 2020. Adversarial augmentation policy search for domain and cross-lingual generalization in reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3723–3738, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Priyanka Sen and Amir Saffari. 2020. What do models learn from question answering datasets? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2429–2438, Online. Association for Computational Linguistics.

Chenglei Si, Ziqing Yang, Yiming Cui, Wentao Ma, Ting Liu, and Shijin Wang. 2021a. Benchmarking robustness of machine reading comprehension models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 634–644, Online. Association for Computational Linguistics.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021b. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1569–1576.

Chao Wang and Hui Jiang. 2019. Explicit utilization of general knowledge in machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2263–2272, Florence, Italy. Association for Computational Linguistics.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 575–581, New Orleans, Louisiana. Association for Computational Linguistics.

Mingzhu Wu, Nafise Sadat Moosavi, Andreas Rücklé, and Iryna Gurevych. 2020. Improving QA generalization by concurrent modeling of multiple biases. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 839–853, Online. Association for Computational Linguistics.

Zhijing Wu and Hua Xu. 2020. Improving the robustness of machine reading comprehension model with hierarchical knowledge and auxiliary unanswerability prediction. *Knowledge-Based Systems*, 203:106075.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. The art of abstention: Selective prediction and error regularization for natural language processing. 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 1040–1051.

Ziqing Yang, Yiming Cui, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. 2019. Improving machine reading comprehension via adversarial training. *arXiv preprint arXiv:1911.03614*.

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-IJCNLP 2021*, pages 1958–1970, Online. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z Sheng, Ahoud Abdulrahmn F Alhazmi, and Chenliang Li. 2019. Generating textual adversarial examples for deep learning models: A survey. *arXiv preprint arXiv:1901.06796*, page 129.

A Generalization Examples

See table 6 for examples.

Some examples in Natural Questions after changing format	
Question	what was the tower of london originally used for
Context	The Tower of London, officially Her Majesty's Royal Palace and Fortress of the Tower of London, is a historic castle located on the north bank of the River Thames in central London. It lies within the London Borough of Tower Hamlets, separated from the eastern edge of the square mile of the City of London by the open space known as Tower Hill. It was founded towards the end of 1066 as part of the Norman Conquest of England. The White Tower, which gives the entire castle its name, was built by William the Conqueror in 1078 and was a resented symbol of oppression, inflicted upon London by the new ruling elite. The castle was used as a prison from 1100 (Ranulf Flambard) until 1952 (Kray twins),[3] although that was not its primary purpose. A grand palace early in its history, it served as a royal residence. As a whole, the Tower is a complex of several buildings set within two concentric rings of defensive walls and a moat. There were several phases of expansion, mainly under Kings Richard I, Henry III, and Edward I in the 12th and 13th centuries. The general layout established by the late 13th century remains despite later activity on the site.
Answer	Text:as a royal residence; Answer_start:794 Text:a royal residence; Answer_start:797
Question	where does the mary river start and finish
Context	The river rises at Booroobin in the Sunshine Coast hinterland, west of Landsborough. From its source, the Mary River flows north through the towns of Kenilworth, Gympie, Tiaro and Maryborough before emptying into the Great Sandy Strait, a passage of water between the mainland and Fraser Island, near the town of River Heads, 17 km (11 mi) south of Hervey Bay. The Mary River flows into the Great Sandy Strait, near wetlands of international significance recognised by the International agreement of the Ramsar Convention and the UNESCO Fraser Island World Heritage Area, which attracts thousands of visitors every year.
Answer	[]

Table 6: Some examples in Natural Questions dataset after using script provided by Sen and Saffari (2020) to change its format into standard SQuAD style.

Trained on clean data		AddKSentDiverse		SQuAD 2.0 dev		Natural Questions	
Feature kind	Feature selection	Acc	F1	Acc	F1	Acc	F1
Baseline(without calibrator)		55.04	53.41	86.39	87.39	59.75	53.30
manual	$c_i + q_i + l_{i0}$	55.04	53.42	85.90	87.29	57.44	52.91
	$+p_{ij}$	55.02	53.64	86.02	87.25	58.41	53.19
	$+p_{ij} + E_i$	55.12	53.67	86.10	87.30	58.44	53.20
	$+sp_{ij}$	**55.32**	**53.91**	85.80	87.20	58.62	53.32
	$+sp_{ij} + E_i$	55.06	53.8	85.87	87.21	58.39	53.18
representation learning	$+v_i$	55.10	53.43	85.31	86.97	59.78	53.32
	$+h_{i,n}$	54.75	53.24	**86.41**	**87.38**	59.78	53.32
	$+A_i$	55.12	53.43	86.31	87.34	59.69	53.32

Table 7: The results on AddKSentDiverse when calibrator only trained on clean original data. All features have been described in section 3. Baseline result is the output of basic model without calibration. Applying manual features to train the calibrator can improve the performance on AddKSentDiverse. Representation learning features just maintain the baseline. Applying the mixture of manual features and representation features has similar results with only apply manual features to train, which we omit in the results.

B Calibrator trained on in-domain data only

Under this setting, the calibrator is only trained on the separated SQuAD 2.0 dataset.

As main experiments, we take manual features and representation learning features described in section 3 into consideration. Accuracy of calibrator, EM and F1-score are the metrics to be evaluated. We take AddKSentDiverse as a representative to demonstrate varying results under different selection of features in table 7.

From the experimental results, we found that manual features can be helpful when calibrator only trained on clean data. It can improve performance of adversarial dataset by 1% while degradation by less than 0.2% on the original dataset. Since the calibrator is ignorant of distribution-shift data, it can't utilize representation learning features and just maintain the baseline result. Among manual features, E_i and sp_{ij} we proposed can be most informative to calibration. It seems that improving the performance of distribution-shift data without sacrificing the original performance is infeasible when calibrator is only trained on the clean data. Further exploration on better features is required.