arXiv:2010.05581v1 [cs.CL] 12 Oct 2020

Counterfactual Variable Control for Robust and Interpretable Question Answering

Sicheng Yu[1], **Yulei Niu**[2], **Shuohang Wang**[3], **Jing Jiang**[1], **Qianru Sun**[1]
[1] Singapore Management University, [2] Nanyang Technological University, [3] Microsoft Dynamics 365 AI Research
scyu.2018@phdcs.smu.edu.sg, yn.yuleiniu@gmail.com, shuowa@microsoft.com, {jingjiang, qianrusun}@smu.edu.sg

Abstract

Deep neural network based question answering (QA) models are neither robust nor explainable in many cases. For example, a multiple-choice QA model, tested without any input of *question*, is surprisingly "capable" to predict the most of correct options. In this paper, we inspect such spurious "capability" of QA models using causal inference. We find the crux is the shortcut correlation, *e.g.*, unrobust word alignment between *passage* and *options* learned by the models. We propose a novel approach called Counterfactual Variable Control (CVC) that explicitly mitigates any shortcut correlation and preserves the comprehensive reasoning for robust QA. Specifically, we leverage multi-branch architecture (Cadene et al. 2019) that allows us to disentangle robust and shortcut correlations in the training process of QA. We then conduct two novel CVC inference methods (on trained models) to capture the effect of comprehensive reasoning as the final prediction. For evaluation, we conduct extensive experiments using two BERT backbones on both multi-choice and span-extraction QA benchmarks. The results show that our CVC achieves high robustness against a variety of adversarial attacks in QA while maintaining good interpretation ability[1].

Introduction

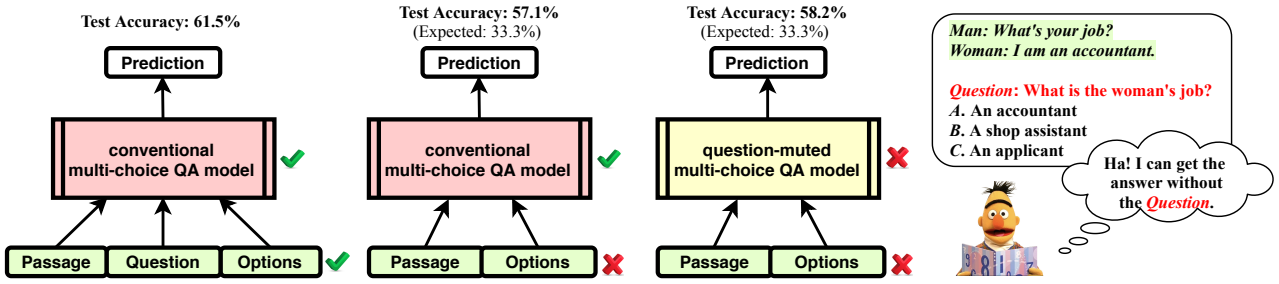
Recently, the error rates on the multiple-choice question answering (MCQA) and span-extraction question answering (SEQA) benchmarks were smashed overnight by large-scale pre-trained models such as BERT (Devlin et al. 2019), XL-Net (Yang et al. 2019), RoBERTa (Liu et al. 2019) and Megatron-LM (Shoeybi et al. 2019). Impressively, using Megatron-LM achieved less than 10% error on RACE (Lai et al. 2017). However, top-performing models often lack interpretability (Feng et al. 2018; Kaushik and Lipton 2018), nor are they robust to adversarial attacks (Ribeiro, Singh, and Guestrin 2018; Szegedy et al. 2013; Wallace et al. 2019). For example, adding one more question mark to the input *question*, which is a simple adversarial attack, may decrease the performance of a QA model (Ribeiro, Singh, and Guestrin 2018). This kind of vulnerability will raise security concerns when the model is deployed in real systems, *e.g.*, intelligent shopping assistant. It is thus desirable to figure out why this happens and how to improve the robustness of the model.

[1]Our code is publicly available on GitHub: https://github.com/PluviophileYU/CVC-QA

In this paper, we carefully inspect the training and test processes for QA models. We find the mentioned vulnerability is caused by the fact that the model often overfits to the *correlation* in training. To illustrate this, we show some example results of BERT-base MCQA model (Devlin et al. 2019) in Figure 1. Specifically, (a) is the conventional result as a reference. (b) and (c) are special results as there is *no question* in the model input. It is surprising that *no question* during test as in (b), or during both training and test as in (c), has very little performance drop, *i.e.*, less than 5 percentage point in accuracy. Our hypothesis is that the BERT-base MCQA model uses a huge amount of network parameters to brutally learn the *shortcut correlation* between the *no question* input (*passage* and *options* only) and the ground truth *answer*. We give an example in Figure 1 (d) to show that this *shortcut* could be achieved by aligning words between the *passage* and the *options*. Can we just conclude that *questions* have little effect on *answers*? We must say no, as we understand this totally violates our common sense about the causality of QA — *the question causes the answer*. Similarly, we may ask what effect the *passage* has.

Driven by these questions in mind, we try to figure out the overall causality of QA in this paper, based on causal inference (Pearl et al. 2009; Pearl and Mackenzie 2018). We begin by analyzing the causal relationships among QA variables, *i.e.*, associating any two variables conditional on the causal effect from one on another. Thanks to the representation methods of causality (Pearl et al. 2009; Qi et al. 2019; Tang et al. 2020; Zhang et al. 2020a), we can represent these QA relationships using the Structural Causal Model (SCM). In Figure 2 (a), we show the SCM we build for MCQA as an example. Each node denotes a variable, *e.g.*, Q for *question*, and the arrow between two nodes denotes a causal relationship, *e.g.*, $Q \rightarrow A$ represents *question causes answer*. It is worth mentioning that R stands for the comprehensive *reasoning* among *passage*, *question* and *options*, which is expected in robust QA.

Based on the SCM, it becomes obvious that not only comprehensive *reasoning* but also shortcut correlations have effects on the prediction results of QA models. For example, as highlighted in Figure 2 (b), P and O can still reach A without Q (the success rate is 24% higher than random guess, as shown in Figure 1 (b)). These shortcut correlations are "distractors" against our goal of robust QA (i.e., the prediction



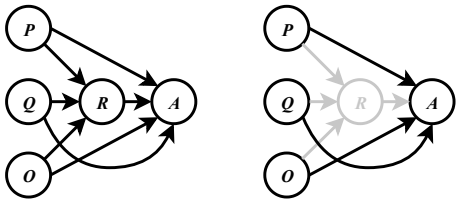
(a) conventional training and test

(b) conventional training but question-muted test

(c) question-muted training and test

(d) a case study of training sample

Figure 1: We observe multi-choice QA models are "capable" to answer a question without any *question* data in input (question-muted) during test (b), or during both training and test (c). We conduct these experiments using the BERT-base model (Devlin et al. 2019) on the multi-choice QA benchmark DREAM (Sun et al. 2019). (a) shows the normal case for reference. (d) show a training sample on DREAM.



(a) SCM

(b) shortcut

Figure 2: The SCM of MCQA. P is for *passage*, Q for *question*, O for *options* and A for *answer*. Particularly, R denotes the comprehensive *reasoning*.

should be caused by only the comprehensive *reasoning*). To alleviate shortcuts, we propose a novel approach called Counterfactual Variable Control (CVC) based on causality theory. CVC in essence includes *counterfactual analysis* (Pearl et al. 2009; Pearl and Mackenzie 2018; Pearl 2001) and *variable control*. The first one imagines a counterfactual world (Roese 1997) where the controlled variable (*e.g.*, Q) had existed to derive its subsequent variables (*e.g.*, R). This avoids any interference on uncontrolled variables that are not input (*e.g.*, R). The second one enables us to explicitly separate comprehensive reasoning and shortcut correlations by variable control, *e.g.*, muting Q in Figure 1 (b) and (c). To implement CVC in deep models, we leverage a multi-branch neural network architecture (Cadene et al. 2019) composed of a robust (comprehensive reasoning) branch and several shortcut branches. We highlight that CVC training exactly follows the conventional supervised training, while CVC testing is based on *counterfactual analysis* to capture the indirect effects (*i.e.*, the predictions from comprehensive reasoning only) for robust QA. In experiments, we validate the efficiency and generality of the proposed CVC approach by using different backbone networks such as BERT-base and BERT-large (Devlin et al. 2019) and experimenting on four QA benchmarks.

Our contributions thus include (i) an overall causality analysis using SCM for QA; (ii) a novel CVC approach to mitigate the shortcut correlations while preserving the robust reasoning in QA; (iii) plugging CVC in different deep backbones and evaluate it on large-scale QA benchmarks.

Related Work

Robustness in NLP. Many recent works generate adversarial examples to augment the training data such as to make the trained more robust against adversarial attacks (Ribeiro, Singh, and Guestrin 2018; Liu et al. 2020; Jia and Liang 2017; Wang and Bansal 2018). They achieve a fairly good performance but their problems are obvious. First, they need to be aware of the prior knowledge of the coming adversarial attack, *i.e.*, "in what way to generate adversarial examples", which is often not available in real applications. Second, their model performance strongly relies on the quality of adversarial examples as well as the training hyperparameters (*e.g.*, augmentation ratios). Alternative methods (for robustness in NLP) include using advanced regularizer (Yeh and Chen 2019), training loss (Jia et al. 2019; Huang et al. 2019), sample filtering (Yaghoobzadeh et al. 2019; Bras et al. 2020) and model ensembling (Clark, Yatskar, and Zettlemoyer 2019; Cadene et al. 2019; He, Zha, and Wang 2019; Utama, Moosavi, and Gurevych 2020). However, it is uncertain why and how these methods can train the QA model to do robust inference. **The relationship and difference between them and our approach are as follows.** In terms of the detailed implementation, our CVC is related to model ensembling. While, we want to highlight that our systematical and explainable causal formulation for QA is the main contribution that potentially opens a principled direction to understanding the real problems of existing QA models and finding out solutions. So, the mentioned model ensembling works can be regarded as implementation-level examples under our formulation.

Adversarial Attacks in QA. Adversarial attack is an indirect way to evaluate the robustness of machine models (Zhang et al. 2020b). The works on NLP adversarial attack can be roughly divided into two categories: word/sentence-level attack and character-level attack. The first category includes text paraphrasing (Ribeiro, Singh, and Guestrin 2018; Zhang, Baldridge, and He 2019; Iyyer et al. 2018) and word sub-

stitution (Ren et al. 2019; Zhang et al. 2019; Alzantot et al. 2018), and the second category is mainly based on character-wise perturbation (Ebrahimi, Lowd, and Dou 2018; Ebrahimi et al. 2018). Our used adversarial attack methods belong to the first category. Besides, most of previous works propose "how to attack" but rarely mention "how to defend". Our work considers both, and importantly, our proposed defending approach is validated robust against unseen attacks (in our experiments).

Causal Inference in Deep Learning. Causal inference (Pearl et al. 2009; Pearl and Mackenzie 2018; Neuberg 2003) is based on the causal assumption make in each specific task, *e.g.*, QA task in this paper. It has been widely applied to epidemiology (Rothman and Greenland 2005), computer science (Van der Laan and Rose 2011), and social science (Steel 2004). Recently, it is incorporated in a variety of deep learning applications such as image classification (Goyal et al. 2019), image parsing (Zhang et al. 2020a), few-shot learning (Yue et al. 2020), long-tail problems (Tang, Huang, and Zhang 2020), pre-training (Wang et al. 2020), scene graph generation (Tang et al. 2020) and vision-language tasks (Qi et al. 2019; Yang, Zhang, and Cai 2020; Chen et al. 2020; Niu et al. 2020; Abbasnejad et al. 2020). In NLP, (Kaushik, Hovy, and Lipton 2020; Lawrence and Riezler 2018) propose causal inference methods used for SNLI (Bowman et al. 2015) and semantic parsing. In contrast, our work makes the first trial to improve the robustness of QA models.

Counterfactual Variable Control (CVC)

Notation. We use MCQA (with its SCM given in Figure 2) as a study case of QA tasks, and introduce the notation for CVC. Basically, we use uppercase letters to denote MCQA variables (*e.g.*, Q for *question*) and lowercase letters for the value of a variable (*e.g.*, q for a specific question). We also introduce the notation of counterfactual values of variables, *i.e.*, the imagined values as if the ancestor variables had existed (uncontrolled) in a counterfactual world (Pearl et al. 2009; Tang et al. 2020; Pearl 2001; Roese 1997). We highlight that our overall notation is general and imposes no constraints on the detailed implementation of QA models. Based on the input variables (with their normal or counterfactual values), we define the notation for the two cases of model prediction: Normal Prediction and Counterfactual Prediction.

Normal Prediction (NP) means the model makes a normal prediction according to realistic input values. We use the function format $Y(X = x)$, abbreviated as Y_x, to represent the effect of $X = x$ on Y. We use this format to formulate any path on the SCM, so we can derive the formulation of normal prediction A as:

$$A_{p,q,o,r} = A(P{=}p, Q{=}q, O{=}o, R{=}r), \quad (1)$$

where $r = R(P = p, Q = q, O = o)$, and $A_{p,q,o,r}$ denotes the inference logits of the model. If all input data are controlled (*e.g.*, by muting their values as null *), we have:

$$A_{p^*,q^*,o^*,r^*} = A(P{=}p^*, Q{=}q^*, O{=}o^*, R{=}r^*), \quad (2)$$

where $r^* = R(P = p^*, Q = q^*, O = o^*)$, and A_{p^*,q^*,o^*,r^*} is the inference logits of the model with null values of input variables (denoted as p^*, q^*, o^*).

Counterfactual Prediction (CP) means the model makes the prediction when some variables are controlled but the others are assigned with counterfactual values by *imagining a counterfactual world of no variable control*. For example, when we control Q with its input value set to null (denoted as q^*), we can assign Q's subsequent variable R with a counterfactual value r by imagining a world where q had existed and had derived r as its normal value $R(P{=}p, Q{=}q, O{=}o)$ (the input of the other two variables are $Q{=}q, O{=}o$). This is a key operation in the *counterfactual analysis* (Pearl et al. 2009; Pearl and Mackenzie 2018; Pearl 2001).

To conduct CVC inference, we propose two variants of counterfactual control: (i) control only input variables; and (ii) control only mediator variable. For (i), we can formulate A as:

$$A_{p^*,q^*,o^*,r} = A(P{=}p^*, Q{=}q^*, O{=}o^*, R{=}r), \quad (3)$$

where p^*, q^*, o^* denote null (variables are muted). Similarly, for (ii), we have:

$$A_{p,q,o,r^*} = A(P{=}p, Q{=}q, O{=}o, R{=}r^*), \quad (4)$$

where R is the only muted variable.

Using NP and CP for CVC Inference. The idea of CVC is to preserve only the robust prediction which is derived by comprehensive reasoning rather than any shortcut correlations. Thanks to the theory of causality (Morgan and Winship 2015), this CVC can be realized by computing the difference between the normal prediction (NP) and the counterfactual prediction (CP). An intuitive interpretation of this computation is that the importance of a variable can be indirectly revealed by generating the results when imagining this variable had not existed (and then comparing to the real results). If the result difference is not significant, it means this variable is useless, otherwise this variable is important and its pure contribution to the results is exactly the computed difference. As in our case, such imagination can be applied on either input (*e.g.*, Q) or mediator variables (*e.g.*, R), so there are two realizations of CVC respectively corresponding to them.

CVC on Input Variables (CVC-IV) is derived as:

$$\text{CVC-IV} = A_{p^*,q^*,o^*,r} - A_{p^*,q^*,o^*,r^*} \quad (5)$$

where particularly in $A_{p^*,q^*,o^*,r}$ input variables are controlled (to be null) while the mediator variable uses its counterfactual value (generated by imaging a counterfactual world where there had no control on input variables).

CVC on Mediator Variable (CVC-MV) is derived as:

$$\text{CVC-MV} = A_{p,q,o,r} - A_{p,q,o,r^*}, \quad (6)$$

where in A_{p,q,o,r^*} input variables use observed values while the mediator variable is controlled (*i.e.*, by imagining a counterfactual world where all inputs were set to null).

Both CVC-IV and CVC-MV aim to capture the causal effect of comprehensive *reasoning* in QA. Their difference lies in *on which variables to apply the control*. In CVC-IV, it is on the inputs variables (which removes any shortcut correlations), while in CVC-MV, it is on the mediator variable (which preserves only the effect of comprehensive *reasoning* after the subtraction). The idea of CVC-IV is more direct than that of CVC-MV. We will show in experiment that they perform differently in different experimental settings of QA.

The Implementation of CVC

We introduce how to implement CVC in deep neural networks. We have two subsections: one for CVC training and the other for CVC-IV and CVC-MV inferences using trained models. We highlight that CVC training is just the supervised training on multi-task networks which is not novel (Cadene et al. 2019; Clark, Yatskar, and Zettlemoyer 2019), some of works on other tasks using similar architecture, but our CVC-IV and CVC-MV inference methods are derived from our causal analysis of QA models — our main contribution to QA methodology.

Multi-task Training

As illustrated in Figure 3, we deploy the state-of-the-art BERT-based QA model (Devlin et al. 2019), as the baseline, and add a few more network branches each of which handles a case of muting part of variables. We take the main branch having complete variables as input to learn the causal effect corresponding to the robust path of SCM (*i.e.*, $P, Q, O \rightarrow R \rightarrow A$), so we call it comprehensive reasoning branch (or robust branch). We use the other branches taking incomplete inputs (part of variables muted) to explicitly learn the shortcut correlations corresponding to the shortcut paths of SCM (*e.g.*, $P, O \rightarrow A$ as Q is muted). We train the model in the manner of multi-task training, *i.e.*, each branch is optimized using an individual objective (the same objective applied across all branches in our case). Only the robust branch gradients can be propagated to update the bottom shared layers in the backbone. Details are as follows.

Robust branch is denoted as F^r. It has the complete input denoted as $\mathcal{X}$, *e.g.*, the realistic values of *question*, *passage* and *options* in MCQA. Its network body, with parameters denoted as θ^r, consists of a pre-trained backbone (*e.g.*, BERT) and a classifier (*e.g.*, one FC layer). Its prediction can be formulated as:

$$A^r = F^r(\mathcal{X}; \theta^r). \tag{7}$$

In CVC, we need to adjust the prediction A^r using shortcut predictions (given by shortcut branches), following the related work (Cadene et al. 2019). We will elaborate the details and explanations in the paragraph of **Loss Computation**.

Shortcut branches aim to explicitly learn the unrobust correlations between incomplete (controlled) input and the ground truth answer. Each branch is denoted as $F^s_n\,(n=1, 2, \cdots, N)$, and takes a subset of variables (denoted as $\mathcal{X}_n$) as input, setting the other variables as null. Its network, with parameters denoted as θ^s_n, has the same architecture with the robust branch. Its prediction can be formulated as:

$$A^s_n = F^s_n(\mathcal{X}_n; \theta^s_n), \tag{8}$$

where $\mathcal{X}_n \subset \mathcal{X}$. A^s_n is used in two ways: (i) computing the individual QA loss to optimize the n-th shortcut branch; and (ii) adjusting A^r, see following details.

Loss Computation. For the n-th **shortcut branch**, we compute its loss as:

$$\mathcal{L}^s_n = -\sum_i \mathtt{p}_i \log \mathrm{softmax}(A^s_{n,i}), \tag{9}$$

where i denotes the i-th dimension of the prediction, and $\mathtt{p}$ is the ground truth (one-hot).

For the **robust branch**, using A^r to compute the loss cannot guarantee the model to learn only comprehensive reasoning, because backpropagating its loss simply makes the model to learn overall correlations as in the convention QA models. We solve this problem by adjusting A^r using shortcut predictions A^s_n. In this way, we can force A^r to *only preserve the prediction that can never be achieved by shortcuts*, *i.e.*, the comprehensive reasoning prediction with the complete input variables as input. We implement this "adjustment" by using the following function:

$$A^e_i = \sum_n \hat{\mathtt{p}}^r_i \cdot \hat{\mathtt{p}}^s_{n,i}, \tag{10}$$

where $\hat{\mathtt{p}}^r_i = \mathrm{softmax}(A^r_i)$, $\hat{\mathtt{p}}^s_{n,i} = \mathrm{softmax}(A^s_{n,i})$ and i is the dimension index of prediction logits. Here we use probabilities instead of logits because we found negative values in logits may reverse the adjustment (Clark, Yatskar, and Zettlemoyer 2019) and probabilities ensure each item in Eq. 10 are of the same scale. We then use the adjusted result A^e to compute the cross-entropy loss for the robust branch:

$$\mathcal{L}^e = -\sum_i \mathtt{p}_i \log \mathrm{softmax}\big(\sum_n \hat{\mathtt{p}}^r_i \cdot \hat{\mathtt{p}}^s_{n,i}\big). \tag{11}$$

We also find that using Eq. 11 may cause the robust branch to focus on only the learning of hard samples. We resolve this by using two loss variants:

$$\begin{aligned} \mathcal{L}^{e1} &= -\sum_n \sum_i \mathtt{p}_i \log \mathrm{softmax}(\hat{\mathtt{p}}^r_i \cdot \hat{\mathtt{p}}^s_{n,i}), \\ \mathcal{L}^{e2} &= -\sum_n w_n \sum_i \mathtt{p}_i \log \mathrm{softmax}(\hat{\mathtt{p}}^r_i \cdot \hat{\mathtt{p}}^s_{n,i}), \end{aligned} \tag{12}$$

where $w_n = \mathrm{softmax}(\mathcal{L}^s_n) = \frac{\exp(\mathcal{L}^s_n)}{\sum_{m=1}^{m=n} \exp(\mathcal{L}^s_m)}$. w_n is a weight used to explicitly enhance the effect of the n-th shortcut branch on the robust branch. Therefore, we formulate the overall loss used in multi-task training as follows,

$$\mathcal{L}^{all} = \mathcal{L}^e + \sum_n \mathcal{L}^s_n. \tag{13}$$

where $\mathcal{L}^e$ can be replaced with $\mathcal{L}^{e1}$ or $\mathcal{L}^{e2}$.

Counterfactual Inferences

Counterfactual inference is different from the conventional inference as it does not directly uses the model prediction. In this section, we explain how to conduct CVC-IV and CVC-MV inferences given the trained models: robust model F^r as well as shortcut models $\{F^s_n\}_{n=1}^N$.

Following the notation formats of NP and CP in Eq. 3 and Eq. 4 along with the notation of output for each branch in Eq. 7 and Eq. 8, we can denote (i) the prediction of the n-th shortcut branch as $a^s_n = F^s_n(p, o; \theta^s_n)$ and its muted value as $a^{s*}_n = F^s_n(p^*, o^*; \theta^s_n)$; and (ii) the prediction of the robust branch as $a^r = F^r(p, q, o; \theta^r)$ and its muted value as $a^{r*} = F^r(p^*, q^*, o^*; \theta^r)$.

In the CVC-IV inference, we mute all input variables, so we get NP as $A_{a^{s*}_1, \cdots, a^{s*}_N, a^{r*}}$ and CP as $A_{a^{s*}_1, \cdots, a^{s*}_N, a^r}$. Using these to replace the NP and CP in Eq. 5 and taking Eq. 10

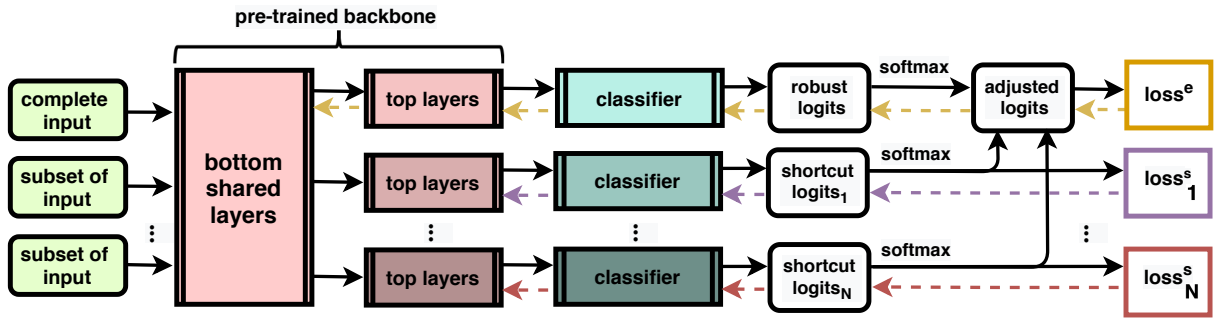


Figure 3: Multi-task training framework in our CVC. The complete input (*e.g.*, $\mathcal{X} = \{P, Q, O\}$ for MCQA) are fed to the robust branch, while a subset to each shortcut (*e.g.*, $\mathcal{X}_n = \{P, O\}$ to the n-th branch). Solid arrows indicate feedforward, and dashed arrows for backpropagation.

into the replaced Eq. 5, we can derive the **CVC-IV inference result** as:

$$\begin{aligned} \text{CVC-IV} &= A_{a_1^{s*},\cdots,a_N^{s*},a^r} - A_{a_1^{s*},\cdots,a_N^{s*},a^{r*}} \\ &= \sum_n \hat{\mathrm{p}}^r \cdot c_n^s - \sum_n c_n^r \cdot c_n^s, \end{aligned} \tag{14}$$

where each element in c_n^r or c_n^s is the same constant in $[0, 1]$. We highlight that CVC-IV inference corresponds to computing Natural Indirect Effect (NIE) in causal inference (Pearl and Mackenzie 2018; Pearl 2001). It is equivalent to the normal inference on the robust model, similar to existing works such as Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019). While, CVC-IV is totally derived from the systematical causal analysis in QA and is thus more explainable than Learned-Mixin which is heuristic.

In the CVC-MV inference, we mute A^r, denoted as a^{r*}. We can denote the NP as $A_{a_1^s,\cdots,a_N^s,a^r}$, and the CP as $A_{a_1^s,\cdots,a_N^s,a^{r*}}$. Using these to replace the NP and CP in Eq. 6 and taking Eq. 10 into the replaced Eq. 6, we can derive the **CVC-MV inference result** as:

$$\begin{aligned} \text{CVC-MV} &= A_{a_1^s,\ldots,a_N^s,a^r} - A_{a_1^s,\ldots,a_N^s,a^{r*}} \\ &= \sum_n \hat{\mathrm{p}}^r \cdot \hat{\mathrm{p}}_n^s - \sum_n c_n^r \cdot \hat{\mathrm{p}}_n^s \end{aligned} \tag{15}$$

which is an indirect way of making inference using only the robust branch. It corresponds to computing Controlled Indirect Effect (CIE) in causal inference (Pearl and Mackenzie 2018; Pearl 2001).

From the empirical results of CVC-MV, we find that the hyperparameter c_n^r makes a clear effect. So we additionally train a c-adaptor (denoted as F_n^c) to generate c_n^r automatically. This can be formulated as:

$$c_n^r = F_n^c(\hat{\mathrm{p}}^r, \hat{\mathrm{p}}_n^s, Distance; \theta_n^c), \tag{16}$$

where $F_n^c(x_1, x_2, x_3; \theta_n^c) = \mathbf{W}_n^2 \tanh(\mathbf{W}_n^1[x_1; x_2; x_3])$, $[;]$ is the concatenation operation, and $\theta_n^c = \{\mathbf{W}_n^1, \mathbf{W}_n^2\}$ are the parameters of c-adaptor. $Distance = \mathbf{JS}[\hat{\mathrm{p}}^r || \hat{\mathrm{p}}_n^s]$, where $\mathbf{JS}$ is Jensen-Shannon divergence (Lin 1991). In experiments, we implement c-adaptor using a two-layer MLP and conduct an ablative study to show its efficiency.

	MCTest	DREAM	RACE	SQuAD
Construction	Crowd.	Exams	Exams	Crowd.
Passage type	Child's stories	Dialogues	Written text	Wikipedia
# of passages	660	6,444	27,933	23,215
# of questions	2,640	10,197	97,687	107,785
# of options	4	3	4	-

Table 1: We conduct MCQA experiments on three datasets, *i.e.*, MCTest (Richardson, Burges, and Renshaw 2013), DREAM (Sun et al. 2019), RACE (Lai et al. 2017), and SEQA experiments on the SQuAD dataset (Rajpurkar et al. 2016). "Crowd.": crowd-sourcing; "-": not applicable.

Experiments

We evaluate the robustness of CVC for both MCQA and SEQA, using a variety of adversarial attacks (Zhang et al. 2020b). Specifically, MCQA aims to predict the correct answer from several input options given a passage and a question. SEQA locates the answer span in a passage given a question. We show the dataset information in Table 1. Below we introduce adversarial attacks, implementation details, an ablation study, a case study and the comparison to state-of-the-arts.

Adversarial Attacks on MCQA. We propose 4 kinds of grammatical adversarial attacks to generate 4 sets of adversarial examples (using original test sets as bases), respectively. `Add1Truth2Opt` (`Adv1`) (and `Add2Truth2Opt` (`Adv2`)): we replace one (and two) of the wrong options with one (and two) of the correct ones borrowed from other samples attached to the same passage. `Add1Pas2Opt` (`Adv3`): We replace one of the wrong options with a random distractor sentence extracted from the passage. Note that this distractor does not overlap with the ground truth option except stop words or punctuation marks. `Add1Ent2Pas` (`Adv4`): We first choose one of the wrong options with at least one entity, and then replace each entity with another entity of the same type. Then, we add this modified option (as a sentence) to the end of the passage.

Adversarial Attacks on SEQA.† We utilize 3 kinds of grammatical adversarial attacks (using the original `Dev` set as basis), denoted as `AddSent` (`Adv1`), `AddOneSent` (`Adv2`)

Dataset	Method	BERT-base						BERT-large					
		Test	Adv1	Adv2	Adv3	Adv4	A.G.	Test	Adv1	Adv2	Adv3	Adv4	A.G.
MCTest	CT (Devlin et al. 2019)	68.9	63.9	59.4	20.2	54.8	-	72.3	70.0	66.8	35.5	57.6	-
	CVC-MV	68.1	69.1	**65.6**	26.8	**61.0**	+6.1%	73.2	74.3	73.5	38.4	68.4	+6.2%
	CVC-IV	**69.4**	**70.0**	65.4	**28.7**	59.9	**+6.4%**	**74.4**	**75.5**	**75.1**	**40.4**	**69.5**	**+7.6%**
DREAM	CT (Devlin et al. 2019)	**61.5**	47.5	39.2	20.9	41.8	-	**65.9**	50.6	43.0	25.6	48.2	-
	CVC-MV	60.1	**49.6**	39.9	23.7	45.6	+2.3%	64.0	51.9	**46.5**	26.3	**51.3**	**+2.2%**
	CVC-IV	60.0	49.2	**40.7**	**25.0**	**47.1**	**+3.1%**	64.5	**52.0**	46.2	**26.6**	51.1	+2.1%
RACE	CT (Devlin et al. 2019)	**64.7**	56.0	50.1	36.6	58.3	-	67.9	61.9	57.9	51.0	61.7	-
	CVC-MV	64.4	56.7	51.7	**39.1**	**59.2**	**+1.4%**	**68.5**	62.6	58.2	**52.0**	**65.7**	+1.5%
	CVC-IV	64.1	**57.0**	**52.2**	38.8	58.6	+1.4%	68.4	**63.1**	**59.1**	51.3	65.1	**+1.6%**

Table 2: Accuracies (%) on three MCQA datasets. Models are trained on original training data. BERT-base and BERT-large are backbones. "A.G." denotes the average improvement over the conventional training (CT) (Devlin et al. 2019) for Adv$\star$ sets.

Method	BERT-base					BERT-large				
	Dev	Adv1	Adv2	Adv3	A.G.	Dev	Adv1	Adv2	Adv3	A.G.
CT (Devlin et al. 2019) (by (Liu et al. 2020))	88.4	49.9	59.7*	44.6*	-	**90.6**	60.2	70.0*	50.0*	-
QAInformax (Yeh and Chen 2019)	**88.6**	54.5	64.9	-	+4.9%	-	-	-	-	-
CVC-MV	87.2	55.7	65.3	51.3	+6.0%	90.2	62.6	**72.4**	52.5	+2.4%
CVC-IV	86.6	**56.3**	**66.2**	**51.5**	**+6.6%**	89.4	**62.6**	71.8	**54.1**	**+2.8%**

Table 3: SEQA F1-measure (%) on the SQuAD Dev set (Test set is not public) and adversarial sets. Models are trained on original training data. BERT-base and BERT-large are backbones. "-": not applicable. "*": our implementation using the public code. "A.G.": our average improvement over the conventional training (CT) (Devlin et al. 2019) for Adv$\star$.

and AddVerb (Adv3). AddSent and AddOneSent released by (Jia and Liang 2017) add distracting sentences to the passage. They can be used to measure the model robustness against entity or noun attacks. In addition, we propose AddVerb for which we hire an expert linguist to annotate the data. Examples are as follows. For the question *"What city did Tesla move to in 1880?"*, AddSent sample could be *"Tadakatsu moved to the city of Chicago in 1881."*, and AddVerb sample could be *"Tesla left the city of Chicago in 1880."*. We use AddVerb to evaluate the model robustness against verb attacks.

Implementation Details.† We deploy the pre-trained BERT backbones provided by HuggingFace (Wolf et al. 2019). Following (Clark, Yatskar, and Zettlemoyer 2019; Grand and Belinkov 2019; Ramakrishnan, Agrawal, and Lee 2018), we perform model selection (*i.e.*, choosing the hyperparameters of training epochs) based on the model performance in the development/test sets on the used dataset.

MCQA-Specific.† MCQA has two shortcut correlations (see Figure 2), *i.e.*, $Q \rightarrow A$ and $P \rightarrow A$ ($O \rightarrow A$ is not discussed here as O is mandatory and can not be muted). We inspect them and notice that the effect from the former one is trivial and negligible compared to the latter. One may argue that Q is an important cue to predict the answer. While the fact is when building MCQA datasets, annotators intentively avoid any easy question-answer data. For example, they include a person name in all options of the *who* question. We thus assume $Q \rightarrow A$ has been eliminated during well-designed data collection and utilize one shortcut branch (muting Q). Eq. 11 and Eq. 12 are equivalent for MCQA ($N=1$ and $w_n=1$).

SEQA-Specific.† Different from MCQA, we propose to manually separate the *question* (Q) of SEQA into corresponding parts: entities & nouns (E); verbs & adverbs (V); and the remaining stop words & punctuation marks (S). Therefore, the SCM of SEQA contains four input variables as P (*passage*), E, V and S. The comprehensive *reasoning* variable R mediates between these four variables and *answer* A. We inspect the empirical effects of all shortcut paths, and build shortcut branches with $N=2$: one branch with E muted, and the other with V muted. We use $\mathcal{L}^{e2}$ to train SEQA models.

Results and Analyses

Table 2 and Table 3 show the overall results for MCQA and SEQA, respectively. Table 4 particularly compares ours to ensembling based methods. Table 5 presents an ablation study on SEQA. Figure 4 shows an case study of MCQA.

Overall Results Compared to Baselines and State-of-the-Art.† From Table 2, we can see that both CVC-MV and CVC-IV can surpass the baseline method (Devlin et al. 2019) for defending against adversarial attacks, *e.g.*, by average accuracies of 7% and 1.5% on MCTest and RACE, respectively. It is worth highlighting the example that CVC-IV on BERT-base gains 8.5% on the most challenging Adv3 set of MCTest. These observations are consistent in the results of SEQA, shown in Table 3. Besides, it is clear that ours outperforms the state-of-the-art QAInformax (Yeh and Chen 2019), *e.g.*, by an average of 1.7% F1-measure (on the same BERT-base backbone). Ours also outperforms model ensembling based methods in most of the datasets, as shown in Table 4. Please note that these methods can be regarded as

	MCTest	DREAM	RACE	SQuAD
DRiFt	1.9	2.5	**1.7**	4.5
Bias Prodcut	5.1	-0.1	1	4.1*
Learned-Mixin	1.8	1.0	1.4	2.1*
CVC-MV(ours)	6.1	2.3	1.4	6.0
CVC-IV(ours)	**6.4**	**3.1**	1.4	**6.6**

Table 4: Comparing to related ensemble-based methods: DRiFT (He, Zha, and Wang 2019), Bias Product and learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019) with BERT-base. (*We show A.G. only, due to the page limits*). We implement related methods by replacing Eq. 10 with their adjustment functions. "*": the design of shortcut branch in original paper is used (one shortcut branch on SQuAD), otherwise our design is used (two shortcut branches on SQuAD).

implementation-level examples under our CVC-IV formulation. From Table 2-4, we also notice that CVC-MV often performs worse than CVC-IV on `Adv*` sets but better on in-domain `Test` (or `Dev`) sets. The possible reason is that the important hyperparameter of CVC-MV c_n^r is learned from in-domain data. Please refer to our supplementary materials where we show that augmenting in-domain data with `Adv*` examples greatly improves the performance of CVC-MV.

Ablative Setting	Dev	Adv1	Adv2	Adv3
(1) w/o first Shct.br.	85.5	52.6	62.5	50.8
(2) w/o second Shct.br.	86.1	**57.7**	66.1	42.1
(3) use $\mathcal{L}^e$	72.4	45.9	54.9	42.6
(4) use $\mathcal{L}^{e1}$	86.5	53.5	63.2	46.7
CVC-IV (ours)	**86.6**	56.3	**66.2**	**51.5**
(5) same $c_{r,n}$	85.7	54.3	64.1	51.0
(6) $c_{r,n} = JS$	85.9	54.3	64.2	51.1
(7) $c_{r,n} = Euc$	86.0	54.4	64.1	51.2
(8) w/o $distance$	86.9	55.3	65.0	51.3
(9) w/o $\hat{p}_r$ and $\hat{p}_n$	84.0	53.2	62.6	49.4
(10) features as input	**87.6**	55.0	64.9	50.9
CVC-MV (ours)	87.2	**55.7**	**65.3**	**51.3**

Table 5: The ablation study on SQuAD (BERT-base). (1)-(3) are ablative settings for multi-task training (using CVC-IV); (4)-(9) are ablative settings related to CVC-MV.

Ablation Study.† In Table 5, we show the SEQA results in 10 ablative settings, to evaluate the approach when: (1) remove the first shortcut branch (*E* muted) from the multi-task training; (2) remove the second shortcut branch (*V* muted) from the multi-task training; (3) use $\mathcal{L}^e$ to replace $\mathcal{L}^{e2}$; (4) use $\mathcal{L}^{e1}$ to replace $\mathcal{L}^{e2}$; (5) set c_n^r to the same constant (tuned in $\{0.2, 0.4, 0.6, 0.8, 1\}$) for all input samples; (6) let $c_n^r = \mathbf{JS}[\hat{p}^r || \hat{p}_n^s]$; (7) let $c_n^r = |\hat{p}^r - \hat{p}_n^s|^2 / 2$ where 2 is the upper bound of the square of Euclidean distance; (8) remove $distance$ item in Eq.16; (9) remove $\hat{p}^r$ and $\hat{p}_n^s$ in Eq. 16; and (10) use the features extracted via top layers to replace $\hat{p}^r$, $\hat{p}_n^s$ and **JS** distance in Eq. 16.

Compared to the ablative results, we can see that our full approach achieves the overall top performance. Two exceptions are (i) a higher score is achieved for `Adv1` if without the second shortcut branch and (ii) a slight higher score for `Dev` if replacing $\hat{p}^r$ and $\hat{p}_n^s$ with features. We think (i) is because the model missing a shortcut branch simply over-fits to the corresponding shortcut correlation (bias). For (ii), a possible reason is that higher-dimensional features (of in-domain data) have stronger representation ability than probability vectors.

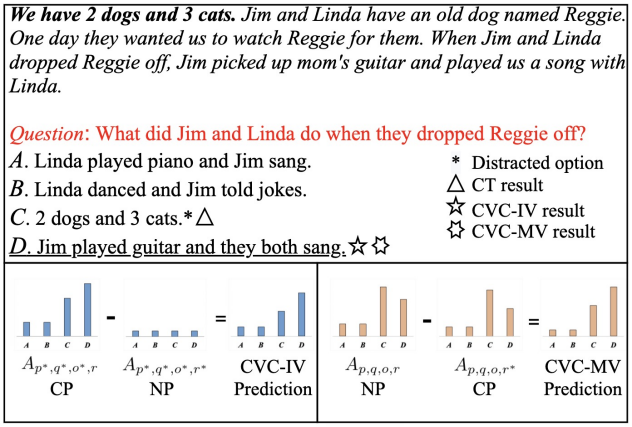


Figure 4: A case study of CVC on MCTest trained on official data. The ground truth is underlined.

Case Study.† We visualize an example of MCQA in Figure 4 to demonstrate the underlying mechanism of CVC-IV and CVC-MV inference. CT (Devlin et al. 2019) merely aligns the words between *passage* and *options*, which leads to the wrong choice C (a confusing choice generated by `Adv1`). In contrast, both our CVC-IV and CVC-MV pick the right answer D. On the bottom blocks, we demonstrate the calculation on prediction logits during CVC-IV (Eq. 5) and CVC-MV (Eq. 6), respectively. We take the CVC-MV as an example to interpret this calculation. Both Normal Prediction (NP) $A_{p,q,o,r}$ and Counterfactual Prediction (CP) A_{p,q,o,r^*} contain the logits of A, B, C and D. The logit value of C is from the word alignment shortcut and it is high in both NP and CP. It thus can be counteracted after the subtraction in CVC-MV. In contrast, the logit value of D is from the comprehensive *reasoning*. When muting the corresponding variable *R* (denoted by r^* in CP A_{p,q,o,r^*}), this value must be reduced. Then it becomes evident after the subtraction in CVC-MV. Please note that we normalize the bar chart (the result of the subtraction) to provide a clear visualization.

Conclusions

We inspect the problem of fragility in QA models, and build the structural causal model to show that the crux is from shortcut correlations. To train robust QA models, we propose a novel CVC approach and implement it on the multi-task training pipeline. We conduct extensive experiments on a variety of QA benchmarks, and show that our approach can achieve high robustness and good interpretation. Our future work is to enhance the structural causal model by considering the subjective factors, *e.g.*, the preference of dataset annotators and the source of passages.

References

Abbasnejad, E.; Teney, D.; Parvaneh, A.; Shi, J.; and Hengel, A. v. d. 2020. Counterfactual vision and language learning. In *CVPR*, 10044–10054.

Alzantot, M.; Sharma, Y.; Elgohary, A.; Ho, B.-J.; Srivastava, M.; and Chang, K.-W. 2018. Generating Natural Language Adversarial Examples. In *EMNLP*, 2890–2896.

Bowman, S.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, 632–642.

Bras, R. L.; Swayamdipta, S.; Bhagavatula, C.; Zellers, R.; Peters, M. E.; Sabharwal, A.; and Choi, Y. 2020. Adversarial Filters of Dataset Biases. *arXiv preprint arXiv:2002.04108* .

Cadene, R.; Dancette, C.; Cord, M.; Parikh, D.; et al. 2019. RUBi: Reducing Unimodal Biases for Visual Question Answering. In *NeurIPS*, 839–850.

Chen, L.; Yan, X.; Xiao, J.; Zhang, H.; Pu, S.; and Zhuang, Y. 2020. Counterfactual Samples Synthesizing for Robust Visual Question Answering. *arXiv preprint arXiv:2003.06576* .

Clark, C.; Yatskar, M.; and Zettlemoyer, L. 2019. Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases. In *EMNLP*, 4060–4073.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 4171–4186.

Ebrahimi, J.; Lowd, D.; and Dou, D. 2018. On Adversarial Examples for Character-Level Neural Machine Translation. In *COLING*, 653–663.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *ACL*, 31–36.

Feng, S.; Wallace, E.; Grissom II, A.; Iyyer, M.; Rodriguez, P.; and Boyd-Graber, J. 2018. Pathologies of Neural Models Make Interpretations Difficult. In *EMNLP*, 3719–3728.

Goyal, Y.; Wu, Z.; Ernst, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. Counterfactual Visual Explanations. In *ICML*, 2376–2384.

Grand, G.; and Belinkov, Y. 2019. Adversarial Regularization for Visual Question Answering: Strengths, Shortcomings, and Side Effects. In *Proceedings of the Second Workshop on Shortcomings in Vision and Language*, 1–13.

He, H.; Zha, S.; and Wang, H. 2019. Unlearn Dataset Bias in Natural Language Inference by Fitting the Residual. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, 132–142.

Huang, P.-S.; Stanforth, R.; Welbl, J.; Dyer, C.; Yogatama, D.; Gowal, S.; Dvijotham, K.; and Kohli, P. 2019. Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation. In *EMNLP*, 4074–4084.

Iyyer, M.; Wieting, J.; Gimpel, K.; and Zettlemoyer, L. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *NAACL*.

Jia, R.; and Liang, P. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*, 2021–2031.

Jia, R.; Raghunathan, A.; Göksel, K.; and Liang, P. 2019. Certified Robustness to Adversarial Word Substitutions. In *EMNLP*, 4120–4133.

Kaushik, D.; Hovy, E.; and Lipton, Z. C. 2020. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. In *ICLR*.

Kaushik, D.; and Lipton, Z. C. 2018. How Much Reading Does Reading Comprehension Require? A Critical Investigation of Popular Benchmarks. In *EMNLP*, 5010–5015.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *EMNLP*, 785–794.

Lawrence, C.; and Riezler, S. 2018. Improving a Neural Semantic Parser by Counterfactual Learning from Human Bandit Feedback. In *ACL*, 1820–1830.

Lin, J. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory* 37(1): 145–151.

Liu, K.; Liu, X.; Yang, A.; Liu, J.; Su, J.; Li, S.; and She, Q. 2020. A Robust Adversarial Training Approach to Machine Reading Comprehension. In *AAAI*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

McCoy, T.; Pavlick, E.; and Linzen, T. 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*, 3428–3448.

Morgan, S. L.; and Winship, C. 2015. *Counterfactuals and causal inference*. Cambridge University Press.

Neuberg, L. G. 2003. Causality: models, reasoning, and inference. *Econometric Theory* 19(4): 675–685.

Niu, Y.; Tang, K.; Zhang, H.; Lu, Z.; Hua, X.-S.; and Wen, J.-R. 2020. Counterfactual VQA: A Cause-Effect Look at Language Bias. *arXiv preprint arXiv:2006.04315* .

Pearl, J. 2001. Direct and indirect effects. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 411–420.

Pearl, J.; and Mackenzie, D. 2018. *The book of why: the new science of cause and effect*. Basic Books.

Pearl, J.; et al. 2009. Causal inference in statistics: An overview. *Statistics surveys* 3: 96–146.

Qi, J.; Niu, Y.; Huang, J.; and Zhang, H. 2019. Two Causal Principles for Improving Visual Dialog. *arXiv preprint arXiv:1911.10496* .

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2383–2392.

Ramakrishnan, S.; Agrawal, A.; and Lee, S. 2018. Overcoming language priors in visual question answering with adversarial regularization. In *NeurIPS*, 1541–1551.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1085–1097.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *ACL*, 856–865.

Richardson, M.; Burges, C. J.; and Renshaw, E. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 193–203.

Roese, N. 1997. Counterfactual thinking. *Psychological Bulletin* 121(1): 133–148.

Rothman, K. J.; and Greenland, S. 2005. Causation and causal inference in epidemiology. *American journal of public health* 95(S1): S144–S150.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053* .

Steel, D. 2004. Social mechanisms and causal inference. *Philosophy of the social sciences* 34(1): 55–78.

Sun, K.; Yu, D.; Chen, J.; Yu, D.; Choi, Y.; and Cardie, C. 2019. DREAM: A Challenge Data Set and Models for Dialogue-Based Reading Comprehension. *TACL* 7: 217–231.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Tang, K.; Huang, J.; and Zhang, H. 2020. Long-Tailed Classification by Keeping the Good and Removing the Bad Momentum Causal Effect. *arXiv preprint arXiv:2009.12991* .

Tang, K.; Niu, Y.; Huang, J.; Shi, J.; and Zhang, H. 2020. Unbiased scene graph generation from biased training. *arXiv preprint arXiv:2002.11949* .

Utama, P. A.; Moosavi, N. S.; and Gurevych, I. 2020. Mind the Trade-off: Debiasing NLU Models without Degrading the In-distribution Performance. *arXiv preprint arXiv:2005.00315* .

Van der Laan, M. J.; and Rose, S. 2011. *Targeted learning: causal inference for observational and experimental data*. Springer Science & Business Media.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. In *EMNLP*, 2153–2162.

Wang, T.; Huang, J.; Zhang, H.; and Sun, Q. 2020. Visual commonsense r-cnn. In *CVPR*, 10760–10770.

Wang, Y.; and Bansal, M. 2018. Robust Machine Comprehension Models via Adversarial Training. In *NAACL*, 575–581.

Williams, A.; Nangia, N.; and Bowman, S. 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, Volume 1 (Long Papers)*, 1112–1122.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; and Brew, J. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771* .

Yaghoobzadeh, Y.; Tachet, R.; Hazen, T. J.; and Sordoni, A. 2019. Robust natural language inference models with example forgetting. *arXiv preprint arXiv:1911.03861* .

Yang, X.; Zhang, H.; and Cai, J. 2020. Deconfounded image captioning: A causal retrospect. *arXiv preprint arXiv:2003.03923* .

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 5754–5764.

Yeh, Y.-T.; and Chen, Y.-N. 2019. QAInfomax: Learning Robust Question Answering System by Mutual Information Maximization. In *EMNLP*, 3361–3366.

Yue, Z.; Zhang, H.; Sun, Q.; and Hua, X. 2020. Interventional Few-Shot Learning. *arXiv preprint arXiv:2009.13000* .

Zhang, D.; Zhang, H.; Tang, J.; Hua, X.; and Sun, Q. 2020a. Causal Intervention for Weakly-Supervised Semantic Segmentation. *arXiv preprint arXiv:2009.12547* .

Zhang, H.; Zhou, H.; Miao, N.; and Li, L. 2019. Generating Fluent Adversarial Examples for Natural Languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5564–5569.

Zhang, W. E.; Sheng, Q. Z.; Alhazmi, A.; and Li, C. 2020b. Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey. *ACM Transactions on Intelligent Systems and Technology* 11(3): 1–41.

Zhang, Y.; Baldridge, J.; and He, L. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *NAACL*, 1298–1308.

Supplementary Materials

These supplementary materials include the entire algorithm, the derivation of loss functions, the generation progress of `AddVerb`, more implementation details and more experiment results for ablation study, case study and data augmentation.

Section A. Algorithm

Algorithm 1 Counterfactual Variable Control (CVC) algorithm

Stage one: multi-task training
Input: complete train set data $\mathcal{X}$ and N different subsets of train set data $\{\mathcal{X}_n\}_{n=1}^N$
Output: F^r with parameters θ^r and $\{F_n^s\}_{n=1}^N$ with parameters $\{\theta_n^s\}_{n=1}^N$

1: **for** `batch` in $\mathcal{X}$ and $\{\mathcal{X}_n\}_{n=1}^N$ **do**
2: **for** n in $\{1, ..., N\}$ **do**
3: optimize θ_n^s with `batch` of $\mathcal{X}_n$ by Eq. 9;
4: **end for**
5: optimize θ^r with `batch` of $\mathcal{X}$ by Eq. 11 for MCQA (by $\mathcal{L}^{e2}$ in Eq. 12 for SEQA);
6: **end for**

Stage two: counterfactual inference
Input: F^r with parameters θ^r, $\{F_n^s\}_{n=1}^N$ with parameters $\{\theta_n^s\}_{n=1}^N$, complete target test data $\mathcal{X}'$ along with its subsets $\{\mathcal{X}'_n\}_{n=1}^N$ and a boolean USE_IV (if $USE\_IV = False$, train set data is also used)
Output: CVC-IV inference result or CVC-MV inference result ($\{F_n^c\}_{n=1}^N$ with parameters $\{\theta_n^c\}_{n=1}^N$)

1: **if** USE_IV **then**
2: compute CVC-IV inference result by Eq. 14;
3: **else**
4: optimize $\{\theta_n^c\}_{n=1}^N$ with $\mathcal{X}$ and $\{\mathcal{X}_n\}_{n=1}^N$ by Eq. 15, Eq. 16 and cross-entropy loss for QA task;
5: compute CVC-MV inference result with target test data $\mathcal{X}'$ and $\{\mathcal{X}'_n\}_{n=1}^N$ by Eq. 15 and Eq. 16;
6: **end if**

This is supplementary to Section "The Implementation of CVC" In Algorithm 1, we summarize the overall process of the proposed Counterfactual Variable Control (CVC) approach. The process consists of two stages: multi-task training (Section 3.1) and counterfactual inference (Section 3.2). Multi-task training aims to train a robust branch F^r and N shortcut branches $\{F_n^s\}_{n=1}^N$. Counterfactual inference performs the robust and interpretable reasoning for QA. $USE\_IV = True$ means using CVC-IV inference and $USE\_IV = False$ means using CVC-MV inference. Line 4 in counterfactual inference can be elaborated as three steps: (1) it uses c-adaptor to compute c_n^r according to Eq. 16 firstly; (2) it derives the logits of CVC-MV inference based on Eq. 15; and (3) it updates c-adaptor using cross-entropy loss (computed between the CVC-MV logits and the ground truth). Line 4 aims to train the c-adaptor with train set data.

Section B. Loss Functions

This is supplementary to Eq. 11 and Eq. 12. Eq. 11 is a straightforward version of loss according to Eq. 10. Here, we prove that Eq. 12 has the same objective as Eq. 11. Assume j is the index of the ground-truth, the equation expansion to Eq. 11 is shown as follows:

$$\begin{aligned}\mathcal{L}^e &= -\sum_i \mathtt{p}_i \log \mathrm{softmax}\big(\sum_n \hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s\big) \\ &= -\log \mathrm{softmax}\big(\sum_n \hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s\big) \\ &= -\log \frac{e^{\sum_n \hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s}}{\sum_i e^{\sum_n \hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s}} \\ &= \log \sum_i e^{\sum_n \hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s} - \sum_n \hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s\end{aligned} \tag{17}$$

Similarly, the equation expansion to Eq. 12 (we use $\mathcal{L}^{e1}$ here since each element in w_n is always positive and we are not able to access to w_n during inference stage) is shown as:

$$\begin{aligned}\mathcal{L}^{e1} &= -\sum_n \sum_i \mathtt{p}_i \log \mathrm{softmax}(\hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s) \\ &= -\sum_n \log \mathrm{softmax}(\hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s) \\ &= -\sum_n \log \frac{e^{\hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s}}{\sum_i e^{\hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s}} \\ &= -\sum_n \big(\hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s - \log \sum_i e^{\hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s}\big) \\ &= \sum_n \log \sum_i e^{\hat{\mathtt{p}}_i^r \cdot \hat{\mathtt{p}}_{n,i}^s} - \sum_n \hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s\end{aligned} \tag{18}$$

From above equations, we see that both Eq. 11 and Eq. 12 aim to maximize $\sum_n \hat{\mathtt{p}}_j^r \cdot \hat{\mathtt{p}}_{n,j}^s$, i.e., the "adjusting" function in Eq. 10.

Section C. AddVerb

This is supplementary to the `AddVerb` in "Adversarial Attack on SEQA". The generation process of `AddVerb` is similar to that of `AddSent` (Jia and Liang 2017). The differences consists of (1) `AddVerb` is used to evaluate the robustness of the model against verb attacks and (2) `AddVerb` instances are annotated by a human expert linguist completely (raw version of `AddSent` is firstly generated by machine). Given a question-answer pair, the linguist creates a distracting `AddVerb` sentence in three steps:

- Replace the verb in the question with an antonym of this verb or an irrelevant verb.
- Create a fake answer with the same type as the ground-truth answer.
- Combine modified question and fake answer, and convert them into the statement.

An illustration of the whole process is shown in Figure 5.

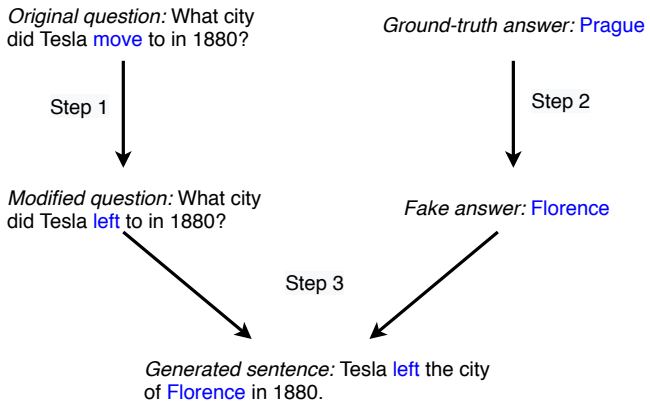


Figure 5: An illustration of the `AddVerb`. We use the instanced question-answer pair in (Jia and Liang 2017) as an example.

Section D. Implementation Details

General Implementation

This is supplementary to "Implementation Details" in Section "Experiments". The overall experiments are conducted on two pieces of Tesla V100 or two pieces of RTX 2080Ti (depending on the useage of memory).

Multi-task Training. The learning rates are fixed to 3e-5 and 2e-5 for BERT-base and BERT-large, respectively. The number of bottom shared layers is fixed to $5/6$ of the total number of layers in the backbone language model for parameter-efficiency, *e.g.*, sharing 10 layers in bottom shared layers when the BERT-base (12 layers) is adopted as the backbone. A maximum sequence length of 384 is adopted in text preprocessing, tokens out of the maximum sequence length are truncated and sequences shorter than 384 will be padded. The number of training epochs is selected amongst $\{16, 24, 36\}$. The batch size is selected amongst $\{16, 24, 32\}$. A linear warm-up strategy for learning rates is used with the first 10% steps in the whole multi-branch training stage. Gradient accumulation and half precision are used to relieve the issue of huge memory usage.

Counterfactual Inference. We use CVC-IV inference for model selection, and apply the chosen checkpoint in both CVC-IV and CVC-MV. The number of epochs in c-adaptor training is selected amongst $\{1, 3, 6, 10\}$ and the batch size of c-adaptor training is fixed to 24. Same learning rates and same warm-up strategy for learning rates in multi-task training are used in c-adaptor training.

MCQA-Specific

This is supplementary to "MCQA-Specific" in Section "Experiments". The results of muting experiments on three MCQA datasets are shown in Table 6. Each number in this table indicates the strength of corresponding direct cause-effects. For example, the results on the row of "No Q" represent the effects of $P \to A$ and $O \to A$ shown in Figure 2 (b). It is clearly shown that the effect of $Q \to A$ is negligible compared to that of $P \to A$. Therefore, we ignore $Q \to A$ when implementing shortcut branches for MCQA. We use one shortcut branch ($N = 1$) with input $\mathcal{X}_1 = \{P, O\}$, to learn $P \to A$ and $O \to A$. Other MCQA-specific implementation details (*e.g.*, how to design the FC layer) are the same with the official code of (Devlin et al. 2019).

	MCTest	DREAM	RACE
Random guess	25.0	33.3	25.0
Complete input	68.9	61.5	64.7
No P	24.2	32.8	41.6
No Q	52.5	57.1	51.0
No P, Q	22.4	33.4	34.7

Table 6: Accuracies (%) of BERT-base MCQA models trained with complete input. "No X" means the value of input variable X is muted.

SEQA-specific

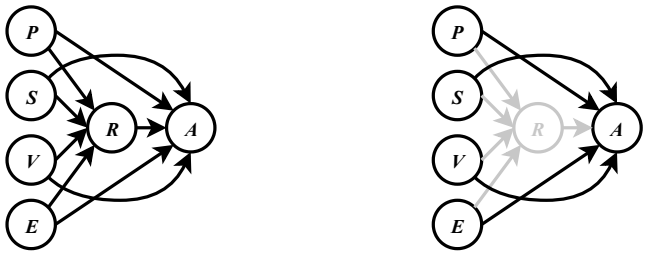


Figure 6: The SCM for SEQA task where Q is decomposed to S, V and E.

	SQuAD
Complete input	88.1
No E	59.4
No V	55.1
No E, V	15.3
No Q	12.4

Table 7: F1 scores (%) of BERT-base SEQA models trained with complete input. "No X" means the value of input variable X is muted.

This is supplementary to "SEQA-Specific" in Section "Experiments".

Partition Details. As mentioned in the main paper, we partition the Q in SEQA into E, V and S. The SCM for SEQA is shown in Figure 6.

- For extracting E in Q, we utilize Stanford part-of-speech tagger and collect the words with the following labels as E: {NN, NNS, NNP, NNPS, CD, SYM, FW}.
- For extracting S in Q, we firstly retrieve the stop words from NLTK and all punctuations, and then remove words

of negation such as "don't" and interrogative words such as "where". We expect to gather less important tokens into *S* (with negligible semantic meanings).

- We group the rest of the words, *e.g.*, verbs and adverbs, and denote them as *V*.

We show the results of muted experiments on SEQA dataset (SQuAD) in Table 7.

Why Conduct This Partition? The reason is simple: *P* is mandatory for SEQA. Without *P* will result in a null prediction of the model. To study the effects of $Q \rightarrow A$, what we can do is to split the variable *Q* into partitions. Besides, our resulting *Q* partitions are intuitive. *E* and *V* contain the most important semantic meanings.

Shortcut Branches. We adopt two shortcut branches ($N = 2$) to represent all shortcut paths in Figure 6(b). The first shortcut branch takes $\mathcal{X}_1 = \{P, S, V\}$ as input and aims to learn $P \rightarrow A$, $S \rightarrow A$ and $V \rightarrow A$. The second shortcut branch takes $\mathcal{X}_1 = \{P, S, E\}$ as input and learns $P \rightarrow A$, $S \rightarrow A$ and $E \rightarrow A$. Other SEQA-specific implementation details (*e.g.*, how to deploy answer pointer layer to make prediction) are the same with the official code of (Devlin et al. 2019).

Section E. More Experimental Results

Overall Results Compared to Baselines and State-of-the-arts.

This is supplementary to "Overall Results Compared to Baselines and State-of-the-arts." in Section "Experiments". We show the full result and implementation details of Table 4 in Table 8 and Table 9.

On MCQA task, we implement DRiFt (He, Zha, and Wang 2019), Bias Product (Clark, Yatskar, and Zettlemoyer 2019) and Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019) following ours CVC implementation (the design of shortcut branch) with their adjustment functions.

On SEQA task, we implement DRiFt by directly changing our adjustment function (Eq. 10) to its. For Bias Product and Learned-Mixin, the corresponding adjustment functions in (Clark, Yatskar, and Zettlemoyer 2019) are used, we also use TF-IDF as the shortcut branch in our experiment which is described in its paper.

From the two tables we can observe that CVC outperforms other methods in most of the cases.

Ablation Study

This is supplementary to "Ablation Study" in Section "Experiments". We conduct the ablation study for MCQA on MCTest with BERT-base. In Table 10, we show the results obtained in 10 ablative settings. Specifically, we (1) use $\mathcal{X}_1 = \{Q, O\}$ as the input of the only shortcut branch; (2) use two shortcut branches, where the first one takes $\mathcal{X}_1 = \{P, O\}$ as input and the second one takes $\mathcal{X}_2 = \{Q, O\}$ as input, and deploy the $\mathcal{L}^e$ in Eq. 11; (3) use the same two shortcut branches as (2), but deploy the $\mathcal{L}^{e1}$ in Eq. 12; (4) use the same two shortcut branches as (2), but $\mathcal{L}^{e2}$ in Eq. 12 is used; (5) set c_n^r to the same constant (tuned in $\{0.2, 0.4, 0.6, 0.8, 1\}$) for all input samples; (6) let $c_n^r = \mathbf{JS}[\hat{p}^r || \hat{p}_n^s]$; (7) let $c_n^r = |\hat{p}^r - \hat{p}_n^s|^2 / 2$ where 2 is the upper bound of the square of Euclidean distance; (8) remove the $distance$ item in Eq. 16; (9) remove $\hat{p}^r$ and $\hat{p}_n^s$ in Eq. 16; and (10) use the features extracted via top layers as input in Eq. 16.

In Table 10, results on (1)-(4) show that considering the shortcut branch with input $\{Q,O\}$ is not effective for the robustness of model. The reason is that this shortcut branch is hard to train, *i.e.*not easy to converge (please refer to "MCQA-specific"). The results suggest: firstly, the shortcut branch with negligible effect magnitude can be ignored (when designing the networks); secondly, if no prior knowledge of the effect magnitude on each shortcut path (of SCM), using $\mathcal{L}^{e2}$ is the best choice. Results on (5)-(10) show the efficiency of our proposed c-adaptor.

Case study

This is supplementary to "Case Study" in Section "Experiments". We visualize the respective examples of CVC-IV and CVC-MV inferences for SEQA. We show the contents of passage, question and predictions for CT, CVC-IV, and CVC-MV in Figure 7.

Specifically, we illustrate CVC-IV and CVC-MV processes for predicting the start token in Figure 8, for simplicity. We note that the process of predicting the end tokens is similar.

Data augmentation

This is supplementary to "Overall Results Compared to Baselines and State of theArt." in Section "Experiments". An intuitive method to improve the model robustness is to augment the training data using adversarial examples (Ribeiro, Singh, and Guestrin 2018; Jia and Liang 2017). We conduct this experiment and show the results of MCTest and present the results of three methods (CT, CVC-IV, and CVC-MV) in Table 11. Comparing Table 11 to the result without data augmentation in main paper, we can see that models get consistently improved via data augmentation. Comparing the results between CT and CVC-*, we find that the latter can achieve further performance boosts for augmented models, *e.g.*, CVC-MV brings an average accuracy of 4% to "Add All" models whose training data are augmented with four kinds of adversarial examples. Please note that we skip the data augmentation experiments for SEQA, because the adversarial attacks (for SEQA) used in our work require a lot of human annotation and proofreading which are costly.

Extend our method to Natural Language Inference (NLI) task

We extend our method to NLI task and compare with other state-of-the-arts methods. Following the setting in previous works, we train the model on MNLI (Williams, Nangia, and Bowman 2018) and evaluate on HANS (McCoy, Pavlick, and Linzen 2019). We use the overlapped tokens in hypothesis and premise as the only bias branch in implementation of CVC. The result is shown in Table 12.

Dataset	Method	Test	Adv1	Adv2	Adv3	Adv4	A.G.
MCTest	CT (Devlin et al. 2019)	68.9	63.9	59.4	20.2	54.8	-
	DRiFt (He, Zha, and Wang 2019)	69.6	66.0	61.9	23.0	54.8	+1.9%
	Bias Product (Clark, Yatskar, and Zettlemoyer 2019)	71.0	66.7	63.6	22.8	65.5	+5.1%
	Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019)	70.5	66.2	60.4	20.2	58.8	+1.8%
	CVC-MV	68.1	69.1	65.6	26.8	61.0	+6.1%
	CVC-IV	69.4	70.0	65.4	28.7	59.9	**+6.4%**
DREAM	CT (Devlin et al. 2019)	61.5	47.5	39.2	20.9	41.8	-
	DRiFt (He, Zha, and Wang 2019)	60.1	48.5	42.2	23.9	44.7	+2.5%
	Bias Product (Clark, Yatskar, and Zettlemoyer 2019)	58.6	47.5	38.8	22.6	40.2	-0.1%
	Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019)	60.9	49.2	41.7	20.0	42.3	+1.0%
	CVC-MV	60.1	49.6	39.9	23.7	45.6	+2.3%
	CVC-IV	60.0	49.2	40.7	25.0	47.1	**+3.1%**
RACE	CT (Devlin et al. 2019)	64.7	56.0	50.1	36.6	58.3	-
	DRiFt (He, Zha, and Wang 2019)	62.0	56.1	53.3	39.3	58.3	**+1.7%**
	Bias Product (Clark, Yatskar, and Zettlemoyer 2019)	62.3	56.7	53.3	37.0	56.8	+1.0%
	Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019)	64.3	56.5	51.9	38.0	60.1	+1.4%
	CVC-MV	64.4	56.7	51.7	39.1	59.2	+1.4%
	CVC-IV	64.1	57.0	52.2	38.8	58.6	+1.4%

Table 8: MCQA accuracies (%) on three datasets. Models are trained on the original training data with BERT-base. "A.G.": our average improvement over the conventional training method (CT) (Devlin et al. 2019) for Adv⋆.

Method	Dev	Adv1	Adv2	Adv3	A.G.
CT (Devlin et al. 2019)	88.4	49.9	59.7	44.6	-
DRiFt (He, Zha, and Wang 2019)	85.7	53.7	65.7	48.5	+4.5%
Bias Product (Clark, Yatskar, and Zettlemoyer 2019)	87.8	53.6	65.7	47.3	+4.1%
Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019)	87.2	53.1	63.9	45.5	+2.1%
QAInformax (Yeh and Chen 2019)	88.6	54.5	64.9	-	+4.9%
CVC-MV	87.2	55.7	65.3	51.3	+6.0%
CVC-IV	86.6	56.3	66.2	51.5	**+6.6%**

Table 9: SEQA F1-measure (%) on the SQuAD Dev set (as Test set is not public) and adversarial sets. Models are trained on the original training data with BERT-base. "A.G.": our average improvement over the conventional training method (CT) (Devlin et al. 2019) for Adv⋆.

Ablative Setting	Dev	Adv1	Adv2	Adv3	Adv4	Average
(1) one modified Shct.br.	68.3	63.1	58.0	24.8	56.5	50.6
(2) two Shct.br. with $\mathcal{L}^e$	70.1	66.8	61.0	24.6	57.1	52.4
(3) two Shct.br. with $\mathcal{L}^{e1}$	70.2	66.7	62.1	25.6	56.5	52.7
(4) two Shct.br. with $\mathcal{L}^{e2}$	70.8	66.6	61.8	27.1	62.2	54.4
CVC-IV (ours)	69.4	70.0	65.4	28.7	59.9	**56.0**
(5) same $c_{r,n}$	68.1	69.3	64.4	25.6	59.3	54.7
(6) $c_{r,n} = JS$	70.1	67.0	61.9	20.8	62.2	53.0
(7) $c_{r,n} = Euc$	69.8	67.7	61.9	22.3	60.5	53.1
(8) w/o $distance$	66.1	67.9	65.2	27.8	61.0	55.5
(9) w/o $\hat{p}_r$ and $\hat{p}_n$	65.6	66.3	64.8	27.4	59.9	54.6
(10) features as input	68.6	68.2	62.9	23.4	59.9	53.6
CVC-MV (ours)	68.1	69.1	65.6	26.8	61.0	**55.6**

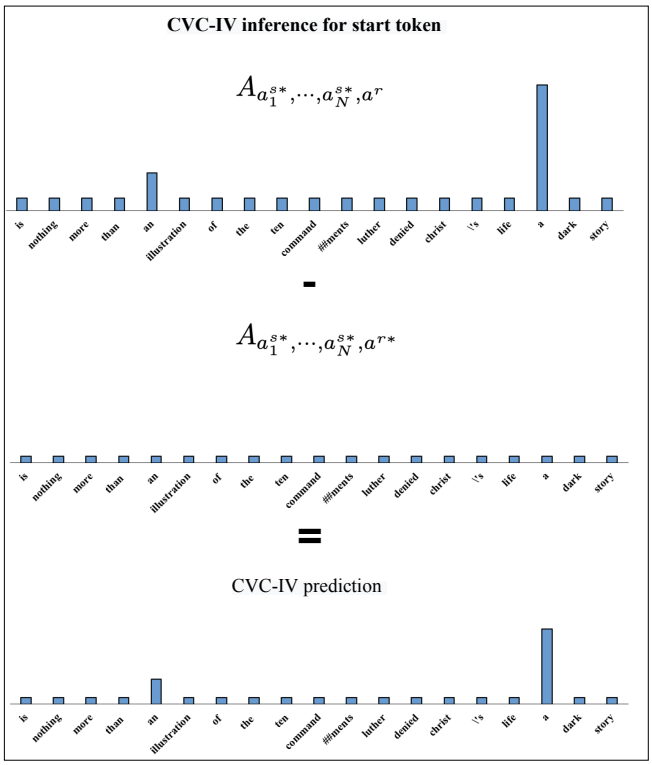
Table 10: The ablation study on MCTest (BERT-base). (1)-(4) are ablative settings for multi-task training (using CVC-IV inference). "Average" means the average performance on Adv⋆ test sets; (5)-(10) are ablative settings related to CVC-MV inference.

On the other hand, Luther also points out that the Ten Commandments when considered not as God's condemning judgment but as an expression of his eternal will, that is, of the natural law also positively teach how the Christian ought to live. This has traditionally been called the "third use of the law." For Luther, also Christ's life, when understood as an example, ***is nothing more than an illustration of the Ten Commandments****, which a Christian should follow in his or her vocations on a daily basis.* ***Luther denied Christ's life a dark story.***

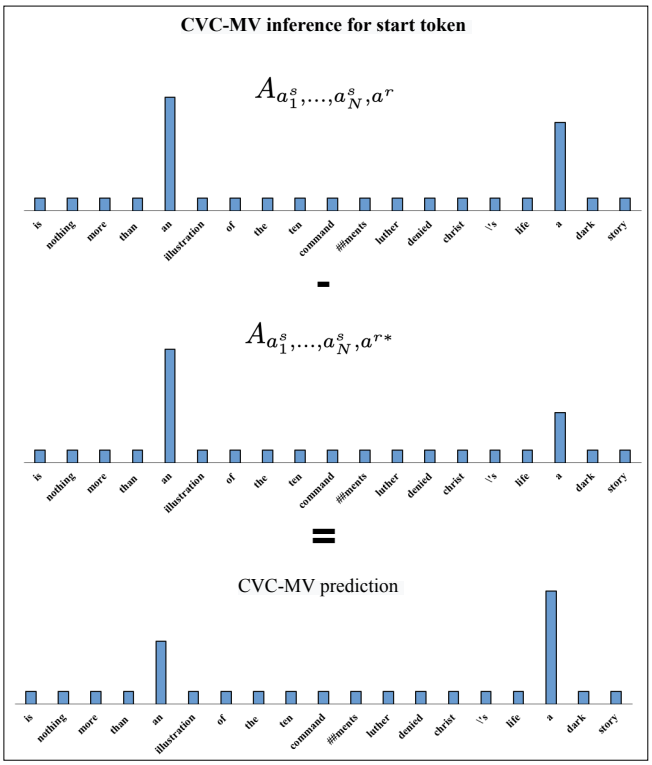
Question: What did Luther consider Christ's life?

Ground-truth answer: *illustration of the Ten Commandments*
CT result: *a dark story*
CVC-IV result: *an illustration of the Ten Commandments,*
CVC-MV result: *an illustration of the Ten Commandments,*

Figure 7: A case study of CVC on SQuAD trained on official data. The distracting sentence from `AddVerb` is underlined.



(a) CVC-IV inference process



(a) CVC-MV inference process

Figure 8: Process of CVC-IV inference and CVC-MV inference corresponding to the case in Figure 7. Only bold tokens in passage are shown, due to limited page size.

		Test	Adv1	Adv2	Adv3	Adv4	AG
	CT	71.0	70.6	72.1	42.5	60.5	-
Add Adv1	CVC-IV	71.7	73.3	74.9	49.2	63.8	+3.9%
	CVC-MV	71.6	72.9	74.8	48.0	62.7	+3.2%
	CT	72.3	73.0	75.1	50.1	63.3	-
Add Adv2	CVC-IV	71.8	73.8	76.2	59.8	65.5	+3.5%
	CVC-MV	71.8	74.2	76.6	61.1	65.5	+3.9%
	CT	67.5	62.7	59.9	70.9	57.1	-
Add Adv3	CVC-IV	67.6	64.5	62.4	70.2	61.6	+2.0%
	CVC-MV	66.8	63.7	62.3	70.3	60.5	+1.5%
	CT	69.8	65.4	60.2	27.7	63.3	-
Add Adv4	CVC-IV	69.9	66.2	62.4	32.7	61.0	+1.4%
	CVC-MV	67.5	65.6	62.4	25.4	66.7	+0.9%
	CT	70.5	72.1	74.1	72.5	63.4	-
Add All	CVC-IV	72.7	73.5	76.4	71.9	68.4	+2.0%
	CVC-MV	**73.1**	**74.6**	**76.6**	**73.3**	**73.5**	**+4.0%**

Table 11: Accuracies (%) on the MCTest dataset, using different kinds of data augmentation in training with BERT-base.

	Matched Dev	HANS
CT	84.2	62.4
Reweight (Clark, Yatskar, and Zettlemoyer 2019)	83.5	69.2
Bias Product (Clark, Yatskar, and Zettlemoyer 2019)	83.0	67.9
Learned-Mixin (Clark, Yatskar, and Zettlemoyer 2019)	84.3	64.0
Learned-Mixin+H (Clark, Yatskar, and Zettlemoyer 2019)	84.0	66.2
DRiFt-HYPO (He, Zha, and Wang 2019)	84.3	67.1
DRiFt-HAND (He, Zha, and Wang 2019)	81.7	68.7
DRiFt-CBOW (He, Zha, and Wang 2019)	82.1	65.4
Mind the Trade-off (Utama, Moosavi, and Gurevych 2020)	84.3	70.3
CVC-IV	82.9	70.0
CVC-MV	83.0	**71.5**

Table 12: NLI accuracies (%) on Matched Dev and HANS. Models are trained on the original training data with BERT-base.