Using counterfactual contrast to improve compositional generalization for multi-step quantitative reasoning

Armineh Nourbakhsh
Language Technologies Institute,
Carnegie Mellon University
J.P. Morgan AI Research
anourbak@cs.cmu.edu

Sameena Shah
J.P. Morgan AI Research
sameena.shah@jpmorgan.com

Carolyn Rosé
Language Technologies Institute,
Carnegie Mellon University
cprose@cs.cmu.edu

Abstract

In quantitative question answering, compositional generalization is one of the main challenges of state of the art models, especially when longer sequences of reasoning steps are required. In this paper we propose CounterComp, a method that uses counterfactual scenarios to generate samples with compositional contrast. Instead of a data augmentation approach, CounterComp is based on metric learning, which allows for direct sampling from the training set and circumvents the need for additional human labels. Our proposed auxiliary metric learning loss improves the performance of three state of the art models on four recently released datasets. We also show how the approach can improve OOD performance on unseen domains, as well as unseen compositions. Lastly, we demonstrate how the method can lead to better compositional attention patterns during training.

1 Introduction

Enterprise documents such as reports, forms, and analytical articles often include quantitative data in tabular form. The data in these tables can be self-contained, but more commonly the surrounding text provides more context that is necessary to understand the content. Answering questions over these hybrid tabular/text contexts requires reasoning that combines verbal and quantitative semantics.

Question answering over quantitative tabular/text data has gained recent traction with the release of datasets such as FinQA (Chen et al., 2021b), TAT-QA (Zhu et al., 2021), and HiTab (Cheng et al., 2022). Table 2 shows an example of a question that requires quantitative reasoning to derive the answer. Given the question and the tabular context, the output is a single-step program that leads to the final answer of `-20`.

A major challenge that state of the art models face is compositional generalization (Montague, 1973), especially when the number of reasoning steps grows (Chen et al., 2021b). In the context of quantitative QA, compositional generalization refers to the model's ability to generalize to new compositions of previously seen elements. As an example, if the model has encountered training examples that demonstrate calculations for "growth rate" and "percent change", we would like it to be able to come up with a reasonable hypothesis as to how to calculate "percent growth" or "rate of change". Table 1 demonstrates how this challenge becomes more difficult as the number of reasoning steps grows. For questions that require longer chains of reasoning, the model learns spurious patterns and unsuccessfully tries to leverage these memorized patterns to solve new problems.

# steps in output	% wrong operator(s)	% wrong operands	% wrong order of operands
1 step	39.07	53.64	7.28
2 steps	46.75	46.75	6.50
3 steps	56.47	29.41	14.12
>= 4 steps	52.00	40.00	8.00

Table 1: Share of FinQANet errors due to the selection of wrong operators or operands when applied to the FinQA dataset (Chen et al., 2021b), broken down by the number of steps in the output. Note that the numbers are based on program accuracy, which accounts for each type of error separately, resulting in the each row summing up to 1.

The Table also shows that as the number of steps grows, generating the wrong operator becomes a more dominant mistake than selecting the wrong operand. Not only is this error more dominant, but it can also have a more destructive impact on the chain of reasoning, as it can derail the model's hidden representations from that point onward. As an example, our analysis of the FinQANet model (Chen et al., 2021b) output showed that if the model generates an incorrect operator, it is about 30% more likely to commit other errors in the following

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 14930–14943

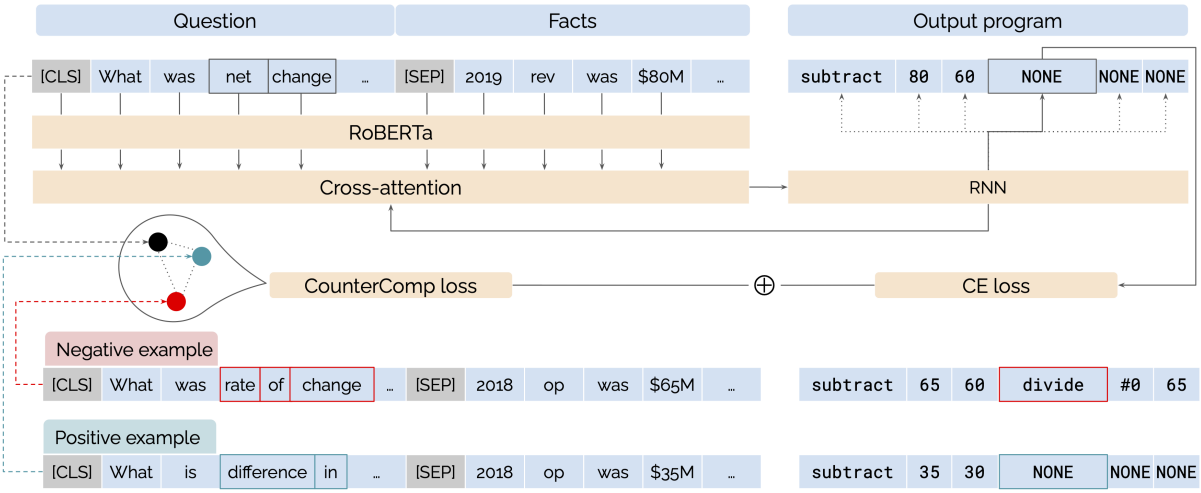

Figure 1: A high-level illustration of our proposed method. The input example (anchor) is processed by cross-attention and recurrent modules to produce the output step by step. In addition to a regular Cross-Entropy loss, CounterComp adds an auxiliary triplet loss based on positive and negative examples. Note that the anchor and pos/neg examples are all processed through the RNN before calculating the triplet loss, a process which we have not illustrated due to space limitations. Also note that multiple pos/neg examples are sampled at each step.

steps compared to when the model generates an incorrect operand.

In this paper, we propose CounterComp, an approach that can enhance compositional learning in multi-step quantitative QA. We take inspiration from the symbolic composition of arithmetic operations, and their correspondence to natural language phrases. Building on the work on attention alignments from previous studies, we propose an auxiliary metric learning loss that is focused on specific components of the input and output. Our sampling strategy is based on counterfactual scenarios. This means that the model learns proper representations for each component based on what-if scenarios. To the best of our knowledge, this is the first study that successfully applies component-wise counterfactual sampling as a metric learning strategy. We show how, when state of the art models are augmented with our auxiliary metric learning loss, they exhibit better performance in cases where multi-step reasoning is required. CounterComp outperforms current baselines on four recently released datasets, and show stronger performance on OOD samples.

2 Related work

The typical architecture of a quantitative QA model is composed of a retriever and a generator (Jurafsy and Martin, 2021). The retriever identifies the particular context where the answer might be found. Since the context can be a mix of table cells and sentences, often a tabular encoder (Herzig et al., 2020) or verbalizer (Chen et al., 2021b) is used to convert the cells into a natural language sequence. The retrieved context is referred to as retrieved *facts*. Next, the generator uses the question along with the facts to generate the output in a step by step fashion. In multi-step QA, the generator often combines a recurrent module with an attention mechanism (Chen et al., 2021b), as illustrated in top half of Figure 1.

Question	What was the net change in revenue from 2019 to 2020?
Tabular context	**Metric ($M)** | **2018** | **2019** | **2020** Operating expenses | 35 | 29 | 30 Revenue | 70 | 80 | 60
Verbalized facts	2019 revenue was $80M. 2020 revenue was $60M.
Output program	`subtract(80, 60)`
Answer	`-20`

Table 2: Example of a quantitative QA problem over tabular data.

The output can be assessed in terms of program accuracy as well as execution accuracy. Our study is focused on improving program accuracy by encouraging compositional generalization in the generator.

There are two common approaches to improving

compositional generalization. Attention alignment models encourage explicit alignments between natural language utterances (e.g. "rate of change") and corresponding symbolic math operations (e.g. subtraction followed by division). Methods informed by counterfactuals use what-if scenarios to generalize to a wider variety of compositions and reduce the effect of memorization.

2.1 Attention alignments

Yin et al. (2021) showed that additional supervision can be used to promote explicit alignments between components in the input and in the output. They added a regularization loss that encourages the cross-attention module to adjust its attention weights according to gold alignments. Using as few as 16 examples, their model was able to improve generalization in a semantic parsing task. CompAQT (Nourbakhsh et al., 2022) extended this idea to multi-step quantitative QA. Instead of using additional supervision, it used natural language heuristics to create noisy alignment labels between input tokens and output symbols. The additional alignment loss improved the performance of three baseline models on multi-step reasoning tasks for four datasets.

2.2 Methods informed by counterfactuals

The success of alignment-based methods is limited by the fact that by heavily discouraging memorization, they underperform in settings where memorization can be helpful (Oren et al., 2020). To strike a balance between memorization and generalization, one approach is to generate new training examples that cover important semantic gaps in the training data. This is reminiscent of how adversarial training can help better define the semantic contours of compositional representations (Zhang et al., 2022). Contrastive or metric learning methods pursue a similar goal, but instead of generating new samples, they leverage existing samples within the training set (Jain et al., 2021).

Counterfactual data augmentation (CAD) methods strive to achieve this by generating new samples using what-if scenarios (Zmigrod et al., 2019; Liu et al., 2021; Chen et al., 2021a). This can be done by altering a minimally sufficient set of tokens in the input such that the output class changes (Kaushik et al., 2020). There are two main challenges to creating these samples. First, it is difficult to identify the minimal set of tokens necessary to alter the output. Second, there is no guarantee that a counterfactual sample exists in the training set. To address these challenges, some studies employ human labelers (Kaushik et al., 2020) or a third party model (Huang et al., 2020). In domains like semantic parsing and quantitative QA where the output is symbolic, an alternative approach leverages the structure of the output to avoid the need for human labelers. Li et al. (2022) achieve this by intervening on the operands. Suppose that a question states "What was the net change in revenue from 2019 to 2020?" and the retriever produces two (verbalized) table cells: "2019 revenue was $80M" and "2020 revenue was $60M". The output program for this question would be: `subtract(80, 60)`. Given the numeric nature of the operands, it's possible to generate new scenarios such as "What if 2019 revenue was $90?" with the updated output `subtract(90, 60)`. Employing this method, Li et al. (2022) augment the TAT-QA dataset (Zhu et al., 2021) into a new dataset named TAT-HQA. They also enhance the verbal reasoning capacity of their model by offering the counterfactual scenario as a natural language prompt. Their model, named Learning to Imagine (L2I), outperforms state of the art models.

As mentioned in the previous section, models that struggle with compositional generalization suffer from errors in operator selection, whereas L2I is focused on the selection of operands. In this paper, we propose CounterComp, a method that focuses on counterfactual sampling for components that indicate operators[1]. Using natural language constraints from previous studies, we first find components that correspond to operators versus those that correspond to operands. Next, we use an auxiliary metric learning loss with positive and negative samples chosen based on those components. This helps us avoid the complexities associated with a data augmentation approach, such as the need for creation of additional human labels. The next section lays out our problem definition in more detail.

3 Problem formulation

Let us consider the example provided by Table 3. Suppose Q is the question, represented as a sequence of tokens $q_1, \cdots, q_N$ (i.e. "what", "was", "the", $\cdots$, "2020", "?").

F is the evidence obtained by the retriever, made up of a sequence of tokens $f_1, \cdots, f_M$ (i.e. "2019",

[1]Please refer to Appendix C for a study on the use of CounterComp for operators versus operands.

"revenue", "was", $\cdots$, "$60M").

The concatenation of these two sequences, i.e. $Q||F$, forms the input to the generator. The generator encodes $Q||F$ using a neural language model such as RoBERTa (Liu et al., 2019), resulting in an embedding matrix $\boldsymbol{U} \in \mathbb{R}^{d_{\text{enc}} \times (N+M)}$.

Consistent with Chen et al. (2021b), we represent the output S as a sequence of steps $s_1, \cdots, s_L$. Each step s_l can be an operator (such as `add` or `divide`), or an operand. Similar to Chen et al. (2021b), our programs are modeled as right-expanding binary trees with each operator having exactly two operands. If necessary, one or more operands are set to `NONE`, where `NONE` is a special constant. L is pre-defined as the maximum number of steps allowed. In the example from Table 3, S is: `subtract`, `80`, `60`, `NONE`, `NONE`, `NONE`.

To generate the lth output step s_l, the generator applies a cross-attention module to $\boldsymbol{U}$, resulting in the attention weight matrix $\boldsymbol{A}_l \in \mathbb{R}^{1 \times (N+M)}$ and the attention output $\boldsymbol{X}_l \in \mathbb{R}^K$. A recurrent module then generates the hidden vector $\mathbf{h}_l$, which is used to produce the output step s_l.

$$\begin{aligned} \mathbf{h}_l &= \text{RNN}(\mathbf{h}_{l-1}, \boldsymbol{X}_l) \\ \mathbf{s}_l &= \text{NN}(\mathbf{h}_l) \end{aligned} \tag{1}$$

where NN can be any neural module that projects $\mathbf{h}_l$ onto the simplex $\mathbf{s}_l \in \mathbb{R}^K$, from which s_l can be sampled: $s_l = \arg\max_k \mathbf{s}_{l,k}$. Our goal is to encourage $\mathbf{h}_l$ to be sensitive to the composition of the input $Q||F$ with regards to the current output step $\mathbf{s}_l$. This means that $\mathbf{h}_l$ needs to capture proper alignments between important terms in the input and the relevant operator/operand in the output.

To achieve this, we pursue a metric learning approach where positive and negative samples are generated according to counterfactual scenarios.

3.1 Counterfactual samples

Given a training example $([Q||F]^{(i)}, S^{(i)})$, we define an *intervention target* $\mathcal{Q}^{(i)}$ as a subsequence of the question tokens, i.e. $\mathcal{Q}^{(i)} = \{q_n^{(i)}; n \in \mathcal{N}^{(i)}\}$ where $\mathcal{N}^{(i)} \subseteq \{1, 2, \cdots, N\}$.

Suppose that changing the intervention target affects a single step in the output program $\mathcal{S}^{(i)} = s_l^{(i)}$, which we name the *intervention outcome*. Note that due to our focus on the generation of operators, we limit the intervention outcome to an operator. Since the output is composed of one operator followed by two operands followed by another operator and so on, l is selected from a limited index set: $l \in \{1, 4, 7, \cdots, L-3\}$. In the example from Table 3, the possible indices will be 1 and 4, representing the operators `subtract` and `NONE`.

Given this definition, it's possible to mine positive and negative examples for the ith training instance. A positive example $([Q||F]^{(i)}_{\text{pos}}, S^{(i)}_{\text{pos}})$ is an instance for which, despite a possible intervention in the target, the outcome remains the same, i.e. $\mathcal{Q}^{(i)}_{\text{pos}} \neq \mathcal{Q}^{(i)}$ and $\mathcal{S}^{(i)}_{\text{pos}} = \mathcal{S}^{(i)}$. A negative example $([Q||F]^{(i)}_{\text{neg}}, S^{(i)}_{\text{neg}})$ is an instance for which an intervention in the target leads to a change in the outcome, i.e. $\mathcal{Q}^{(i)}_{\text{neg}} \neq \mathcal{Q}^{(i)}$ and $\mathcal{S}^{(i)}_{\text{neg}} \neq \mathcal{S}^{(i)}$.

This allows us to define a triplet loss that encourages $\mathbf{h}^{(i)}_l$ to remain close to $\mathbf{h}^{(i)}_{l,\text{pos}}$ and far from $\mathbf{h}^{(i)}_{l,\text{neg}}$ with a margin of $\alpha^{(i)}$:

$$\begin{aligned} &\mathcal{L}^{(i)}_{\text{triplet}} = \\ &\max\{||\mathbf{h}^{(i)}_l - \mathbf{h}^{(i)}_{l,\text{pos}}||_2^2 - ||\mathbf{h}^{(i)}_l - \mathbf{h}^{(i)}_{l,\text{neg}}||_2^2 + \alpha^{(i)}, 0\} \end{aligned} \tag{2}$$

Figure 1 illustrates the sampling process for one training example. Note that this metric learning approach will only be valid if causal assumptions with regards to the intervention target are valid, i.e. the change in $\mathcal{S}^{(i)}_{\text{neg}}$ is in fact the result of the intervention in $\mathcal{Q}^{(i)}_{\text{neg}}$ and not a change in any other part of the input. In a data augmentation setting, this can be achieved by keeping the input fixed and perturbing a small segment that functions as the intervention target similar to (Kaushik et al., 2020). However, as discussed in Section 2.2. this requires additional manual labor to annotate the perturbed examples.

In the next section, we describe how we impose certain constraints on the intervention target to achieve this in a self-supervised setting[2].

4 Methodology

Our goal is to identify potential positive and negative samples for the anchor $([Q||F]^{(i)}, S^{(i)})$. Suppose the anchor is the one shown in the top three rows of Table 3. ***Bold italicized*** tokens are redundant between the question and the fact, e.g. "revenue", "2019", and "2020". Those terms are often used by the retriever to find the correct facts. They are also used by the generator to find the correct order of operands.

[2]Note that the term "self-supervised" is used in this context to refer to the sampling strategy, i.e. no additional labeling is needed to generate the positive and negative samples.

There are also terms that are unique to the question, i.e. "What", "the net change to", "from", and "to" (highlighted in blue). In CompAQT, the authors showed that these can be used as indicators for the operators. Lastly, there are terms that are unique to the facts, i.e. "was $80M" and "was $60M" (red italicized tokens). These can be used as indicators for the operands. We use these heuristics to guide our sampling strategy.

We flag all spans in the question that do not overlap with the facts, i.e. underlined blue segments. Those spans serve as candidate intervention spans. In the example from Table 3, this results in four candidates: "What", "the net change in", "from", and "to".

Next, we seek a positive and a negative example within the training set. A positive example is a sample in which, despite possible changes in the question, the operators in the output remain consistent with the operators in the anchor. Table 3 shows one such example. Several terms have been altered in the question. However, we would only focus on the changes in the candidate spans. Here, "was" has changed to "is", "net change" has changed to "difference", "from" to "between" and "to" to "and". This results in a token-level Levenshtein distance of 5 (four edits and one insertion) (Yujian and Bo, 2007). We ignore the change from "revenue" to "operating expenses" and from "2019" to "2018", because those changes have occurred outside of our candidate spans and only correspond to operands.

A negative example is a sample in which exactly one output operator is altered, deleted, or added. Table 3 shows one such example. Here, the output includes a new operator `divide`. The question has also been altered with a token-level Levenshtein distance of 4.

The given positive and negative example can now be plugged into Equation 2. Instead of a fixed margin, we use the edit distances mentioned before to dynamically adjust the margin. Let $\text{NLD}^{(i)}_{\text{pos}}$ and $\text{NLD}^{(i)}_{\text{neg}}$ be the *normalized, token-level Levenshtein edit distance* between the anchor and the positive example, and the negative example, respectively. We set the margin to: $\alpha^{(i)} = 1 - |\text{NLD}^{(i)}_{\text{neg}} - \text{NLD}^{(i)}_{\text{pos}}|$

This encourages a larger margin for cases where the anchor is equally similar to the positive and the negative examples, and the model might have a harder time picking up on the nuances of each component.

Anchor	**Question**	What ***was*** the net change in ***revenue*** from ***2019*** to ***2020***?
	Facts	***2019 revenue was*** *$80M.* ***2020 revenue was*** *$60M.*
	Program	`subtract`(*80*, *60*) `NONE(NONE, NONE)`
	Candidate intervention spans	What the net change in from to
Positive sample	**Question**	What is the difference in ***operating expenses*** between ***2018*** and ***2020?***
	Facts	***2018 operating expenses*** *were $35M.* ***2020 operating expenses*** *were $30M.*
	Program	`subtract`(*35*, *30*) `NONE(NONE, NONE)`
Negative sample	**Question**	What ***was*** the rate of change of ***operating income from 2018*** to ***2019***?
	Facts	***2018 income from operating*** *activities* ***was*** *$65M.* ***2019 income from operating*** *activities* ***was*** *$60M.*
	Program	`subtract`(*65*, *60*) `divide`(`#0`, *65*)
Variables	$\mathcal{Q}^{(i)}$ $\mathcal{S}^{(i)}$	$\{q_1^{(i)}, q_3^{(i)}, q_4^{(i)}, q_5^{(i)}, q_6^{(i)}, q_8^{(i)}, q_{10}^{(i)}\}$: what the net change in from to $s_4^{(i)}$: NONE
	$\mathcal{Q}^{(i)}_{\text{pos}}$ $\mathcal{S}^{(i)}_{\text{pos}}$ edit dist	$\{q_1^{(i)}, q_2^{(i)}, q_3^{(i)}, q_4^{(i)}, q_5^{(i)}, q_8^{(i)}, q_{10}^{(i)}\}$: what is the difference in between and $s_4^{(i)}$: NONE 5
	$\mathcal{Q}^{(i)}_{\text{neg}}$ $\mathcal{S}^{(i)}_{\text{neg}}$ edit dist	$\{q_1^{(i)}, q_3^{(i)}, q_4^{(i)}, q_5^{(i)}, q_5^{(i)}, q_6^{(i)}, q_{12}^{(i)}\}$: what the rate of change of to $s_4^{(i)}$: `divide` 4

Table 3: Example of positive and negative sampling using counterfactual components. Blue underlined text indicates components that are unique to the question (candidates for intervention). These terms often indicate an operator. *Red italicized* text indicates terms that are unique to the facts. These terms often indicate operands. ***Bold italicized text*** indicates terms that are shared between the question and facts. These terms often indicate metrics.

4.1 Runtime optimization

There are two runtime challenges to this proposed approach: 1) Sampling can be costly if the entire training set has to be scanned for each batch. This means an online sampling strategy cannot be used. On the other hand, an offline strategy introduces a large overhead. A hybrid approach is needed. 2) Calculating the edit distance metric is a costly operation with $O(n^2)$ steps.

To solve the first problem, we build two indices prior to training. One index groups the samples by their sequence of output operators. This index can be used to sample positive examples.

The other index includes all training examples, and for each example, it includes the full list of one-step perturbations applied to its output operators. By generating all possible perturbations, we are able to find other samples whose outputs match the perturbed sequence (i.e. negative samples). For

a sequence with n operators, all possible perturbations can be generated in $O(n \times K)$ time, where K is the number of possible operators[3].

Given the pre-generated positive and negative pools, we can also calculate and cache edit distances ahead of time. However, in practice, we realized that we could do so during training with little additional cost. This is because the edit distance is applied at the token-level[4], and is limited to candidate spans, rendering it relatively fast. The decision as to whether distances should be cached or calculated on the fly depends on the average size of each pool versus the number of training steps.

The algorithm outlined in Appendix B summarizes our approach.

5 Experiments

5.1 Datasets

We use the hybrid CompAQT dataset, which is composed of four previously released datasets, namely **FinQA** (Chen et al., 2021b), **TAT-QA** (Zhu et al., 2021), **HiTab** (Cheng et al., 2022), and **MULTIHIERTT** (Zhao et al., 2022). The authors filtered these four datasets down to QA pairs that require single or multi-step quantitative reasoning. They also processed the tables and outputs in all four datasets to match the FinQA format.

5.2 Baselines

We apply our proposed auxiliary loss to three baselines: 1) **FinQANet**, originally developed for the FinQA dataset (Chen et al., 2021b). 2) **TAGOP**, originally developed for the TAT-QA dataset (Zhu et al., 2021). 3) **Pointer-Verbalizer Network** (PVN), originally proposed by Nourbakhsh et al. (2022). We also apply the **CompAQT** loss to each model as a secondary baseline in order to determine how CounterComp compares to an attention-alignment strategy.

5.3 Sampling success rate

Another possible concern is that our sampling strategy might be limited, in that positive and negative samples might not always be available in the training set, or that limited availability of samples might bias the training process. To remediate the problem of unavailable samples, when a positive sample is missing, we use the anchor as the positive sample, and when a negative sample cannot be found, we use a uniformly sampled instance from the batch.

Table 4 shows some statistics about the success rate of the sampling algorithm. "% Failure" identifies the share of training examples for which either a positive or a negative example was missing. Unsurprisingly this never happens for single-step programs, is very rare for two-step programs, and with the exception of HiTab, happens in less than 10% of the cases for longer programs. The Table also shows the average number of positive and negative examples found for each anchor. Again, HiTab has the lowest number of available samples, making it the most challenging dataset. In Section 6.4, we demonstrate how, even in cases with few possible samples, the model is able to generalize to unseen examples.

5.4 Settings

Since we are focused on the generator, in the experiments discussed in this section we will use gold facts and encode the input using RoBERTa-large (Liu et al., 2019)[5]. We run the baselines with and without the additional $\mathcal{L}_{\text{triplet}}$ for 50 epochs with a learning rate of $5\text{e}{-5}$, the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. At each step, we sample (with replacement) 5 positive and negative pairs per anchor, and add the average auxiliary triplet loss to the main model loss with a weight of λ. After a grid search with a step-size of 0.1, we set λ to 0.4 for all experiments. All experiments were conducted on 8 NVIDIA T4 GPUS with 16 GBs of memory.

6 Results and analysis

Table 5 shows the program accuracy of baselines (top row of each cell) compared the addition of CounterComp loss (bottom row of eachcell). Among the baseline models, TAGOP is not designed to generate multi-step programs. Therefore we only apply it to the TAT-QA dataset, which has a set of pre-determined operations (e.g. `change ratio`). We also apply the PVN model to the combined dataset, but since FinQANet outperforms it on all benchmarks, we will continue to use FinQANet as the reference baseline model for the remaining experiments in this section.

As Table 5 shows, CounterComp consistently

[3]Since we follow Chen et al. (2021b), in all of our experiments $K = 10$.

[4]Since we're using a language model that uses word-piece tokenization, in effect the runtime is at subword level.

[5]Please refer to Appendix A for results using retrieved facts.

Dataset	1 step			2 steps			3+ steps		
	% Failure	Avg. # pos samples	Avg. # neg samples	% Failure	Avg. # pos samples	Avg. # neg samples	% Failure	Avg. # pos samples	Avg. # neg samples
FinQA	0	1457	3254	0.2	913	630	8.2	41	190
TAT-QA	0	1808	638	0	295	958	1.3	1055	66
HiTab	0	221	554	2.5	30	29	29.8	4	24
MULTIHIERTT	0	189	533	0.2	326	335	7.8	92	190

Table 4: The failure rate of sampling from each dataset (when no positive or no negative sample can be found for a given anchor), as well as the average number of positive and negative samples found for each anchor.

outperforms the baselines and the margin is often higher for longer programs. One notable exception is the TAT-QA dataset. As mentioned before, the dataset is not designed for open-ended multi-step reasoning and includes a limited set of possible operations. Therefore methods that encourage memorization might achieve higher performance on TAT-QA. HiTab is another challenging dataset, but despite low performance on longer sequences, CounterComp offers an improvement over the baseline.

Model	Dataset	Program accuracy			
		1 step	2 steps	3+ steps	Overall
TAGOP	TAT-QA	45.01	39.56	42.73	43.25
+CompAQT		46.07	40.28	43.73	43.88
+CounterComp		**46.12**	**41.51**	**45.67***	**45.38**
PVN	Combined	68.14	61.33	13.54	56.64
+CompAQT		70.78	63.45	16.63	59.21
+CounterComp		**71.58***	**64.31***	**18.44***	**61.20***
FinQANet	FinQA	75.63	65.87	30.36	68.44
+CompAQT		78.68	75.12	35.85	73.74
+CounterComp		**79.13***	**75.45***	**36.86***	**74.49***
FinQANet	TAT-QA	**73.33**	63.76	64.88	**70.71**
+CompAQT		70.00	63.76	66.26	69.97
+CounterComp		70.56	**63.80**	**66.90**	70.01
FinQANet	HiTab	34.70	25.14	15.91	30.12
+CompAQT		34.73	29.94	17.35	32.23
+CounterComp		**34.94**	**30.00***	**17.39**	**32.61***
FinQANet	MULTIHIERTT	38.99	40.07	15.01	38.94
+CompAQT		38.87	42.35	16.77	40.82
+CounterComp		**39.25***	**42.51***	**16.86**	**40.85***
FinQANet	Combined	65.11	62.00	30.76	58.60
+CompAQT		66.60	65.14	34.16	60.88
+CounterComp		**67.91***	**66.00***	**36.91***	**61.82***
(Fixed margin)		66.19	64.89	34.06	59.58

Table 5: Program accuracy for program generation using baseline models v.s. using CompAQT loss, v.s. using CounterComp loss. * indicates that a gain/loss is significant at $p < 0.005$ compared to the baseline, using the paired-bootstrap test proposed by Berg-Kirkpatrick et al. (2012) for $b = 10^3$.

6.1 Auxiliary triplet loss versus auxiliary attention alignment loss

The middle row of each cell in Table 5 shows the program accuracy when CompAQT loss is added instead of CounterComp loss. As previously described, CompAQT imposes an auxiliary attention alignment loss such that tokens related to operators receive more attention during the generation of operators. Even though this leads to improvements over the baselines, CounterComp outperforms CompAQT in all experiments. This might be due to the fact that the regularizing effect of CompAQT loss is not as strong as the representation learning impact of CounterComp.

Despite the fact that CounterComp was not designed as an attention alignment model, it does have an impact on how attention patterns evolve during training. Table 6 shows the top-attended input tokens during the generation of a `divide` operator in various contexts. For a singular division operation, FinQANet attends to tokens such as "year" whereas CounterComp encourages the model to attend to more relevant tokens such as "net" and "change". A subtraction followed by a division often indicates a percentage calculation, as captured by both models. An addition followed by a division often indicates an average calculation. Again, CounterComp is able to capture relevant tokens but the FinQANet baseline seems to attend to some memorized tokens such as "annual".

Model	Top attended tokens during the generation of `divide`		
	`divide`	`subtract` **`divide`**	`add` **`divide`**
FinQANet	share, year	ratio, percent	annual, per
+CounterComp	net, change	share, percent	average, per

Table 6: Top attended tokens during the generation of the division operator in various sequences. The dataset used for this experiment is FinQA.

6.2 Fixed versus adaptive margin

The last row of Table 5 shows the performance of the FinQANet model on the combined dataset using the CounterComp loss with a fixed margin of 1. The performance suffers, especially as the number of steps grows. This further demonstrates the importance of the adaptive margin $\alpha^{(i)}$ that takes the edit distance into account.

Question	Evidence	Gold program	FinQANet	FinQANet + CounterComp
What was the gross margin decline in fiscal 2004 from 2003?	1) the gross margin pct of 2004 is 27.3% 2) the gross margin pct of 2003 is 27.5%	`subtract(27.5, 27.3)`	`subtract(27.5, 27.3), divide(#0, 27.5)`	`subtract(27.5, 27.3)`
What percentage of amounts expensed in 2009 came from discretionary company contributions?	1) amounts expensed for 2009 was $35.1 2) expense includes a discretionary company contribution of $3.8	`divide(3.8, 35.1), multiply(#0, const_100)`	`divide(3.8, 35.1)`	`divide(3.8, 35.1), multiply(#0, const_100)`
Did the share of securities rated aaa/aaa increase between 2008 and 2009?	1) the aaa/aaa share of 2009 is 14% 2) the aaa/aaa share of 2008 is 19%	`greater(14, 19)`	`subtract(14, 19), subtract(14, #0)`	`greater(14, 19)`
What is the amount of credit lines that has been drawn in millions as of year-end 2016?	1) We have other committed and uncommitted credit lines of $746 million 2) $554 million of these credit lines were available for use as of year-end 2016	`subtract(746, 554)`	`multiply(554, const_1000000)`	`multiply(746, 554)`

Table 7: Four examples from the FinQA dataset, showing CounterComp's success and failure in capturing compositional expressions. Note that some numbers have been truncated to save space.

Model	Program accuracy on test dataset			
	TAT-QA	HiTab	MULTIHIERTT	FinQA (unseen programs)
FinQANet	41.64	22.80	35.33	65.74
+CompAQT	39.88	22.71	35.28	70.32
+CounterComp	**42.00**	**22.97**	**36.94**	**73.53**

Table 8: OOD performance of FinQANet variations when trained on the FinQA dataset and tested on other datasets, or tested on unseen operator compositions in the FinQA dev set.

6.3 Qualitative examples

Table 7 shows four qualitative examples from the FinQA dataset. The first two rows show how CounterComp enables the FinQANet model to represent concepts such as "decline" and "percentage" more accurately. The third example shows how CounterComp is able to determine the difference between a calculation question and a yes/no question. The last row shows a failure example that was also reported in Chen et al. (2021b). Here, CounterComp does not improve the performance of FinQANet. This particular example requires domain expertise to address, which goes beyond a direct mapping between components in the question and the evidence. This highlights the need for methods that allow domain expertise to be represented more effectively (Chen et al., 2021b).

6.4 Compositional v.s. OOD generalization

In a recent study Joshi and He (2022) showed that current approaches to counterfactual data augmentation do not necessarily lead to better generalization to out-of-distribution (OOD) samples. To test whether this holds for CounterComp, we conduct two studies. First, we train FinQNet with and without CounterComp loss on the FinQA dataset, then test it on the other three datasets. Note that the four datasets are based on different domains. FinQA and TAT-QA are both based on financial reports, but while FinQA was derived from US filings, TAT-QA is based on international filings and therefore covers a wider variety of metrics. HiTab and MULTIHIERTT are both based on other types of corporate reports with highly complex tabular structures.

The first three columns of Table 8 show a slight improvement when CounterComp is used in this setting. In contrast, using CompAQT loss slightly hurts the performance, demonstrating CounterComp's higher OOD generalization potential.

Next, we select a subset of samples from the FinQA dev set that have unseen compositions compared to the training set. This means that the particular combination of operations were never seen during training. As the last column of the table shows, CounterComp outperforms the baseline by more than 7 points. This further demonstrates how improving representation learning at the component level can enhance generalization to unseen contexts.

7 Conclusion

In this paper, we presented CounterComp, a method that leverages counterfactual contrast to enable metric learning for quantitative QA. We show how using the auxiliary CounterComp loss can improve compositional generalization in multi-step reasoning tasks, especially as the number of steps grows.

Due to runtime challenges, we proposed a hybrid offline/online sampling strategy that uses predefined indices for easier lookup operations. This allows us to capture samples that have a contrast of one operator with the anchor. In future studies, we hope to capture contrastive samples with longer perturbation chains. We also hope to examine the effectiveness of counterfactual compositional contrast in other domains such as semantic parsing and question answering over multimodal input.

Lastly, we hope to extend the use of CounterComp to enhance the performance of the retriever, using the heuristics introduced in Section 4 (i.e. by

focusing on components in the question that overlap with the facts). This can result in a quantitative QA pipeline that is powered by compositional contrast in an end-to-end fashion.

8 Limitations

As previously mentioned, our study is focused on the generator component of a QA pipeline and ignores the retrieval task. In the experiments presented in the paper, we have used gold facts to report the results. For certain datasets such as HiTab and MULTIHIERTT which were designed for complex tabular structures, this might simplify the end-to-end challenge. In future studies, we hope to explore whether CounterComp can enhance the performance of retrievers.

The datasets used in our experiments were curated using enterprise documents such as financial reports or other corporate disclosures. Quantitative QA over these reports often involves multi-step reasoning that is limited to linear arithmetic operations such as addition, division, averaging, etc. A completely open-domain QA engine might need to cover more complex operators.

Lastly, we designed CounterComp to leverage existing data by sampling from the training set. Nevertheless, combining CounterComp with augmentation-focused methods such as CAD might lead to more robust models.

References

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Hao Chen, Rui Xia, and Jianfei Yu. 2021a. Reinforced counterfactual data augmentation for dual sentiment classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 269–278, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021b. Finqa: A dataset of numerical reasoning over financial data. *Proceedings of EMNLP 2021*.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2020. Reducing sentiment bias in language models via counterfactual evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 65–83, Online. Association for Computational Linguistics.

Paras Jain, Ajay Jain, Tianjun Zhang, Pieter Abbeel, Joseph Gonzalez, and Ion Stoica. 2021. Contrastive code representation learning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5954–5971, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nitish Joshi and He He. 2022. An investigation of the (in)effectiveness of counterfactually augmented data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3668–3681, Dublin, Ireland. Association for Computational Linguistics.

Dan Jurafsy and James H. Martin. 2021. *Speech and language processing*, 3 edition, chapter 23.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Moxin Li, Fuli Feng, Hanwang Zhang, Xiangnan He, Fengbin Zhu, and Tat-Seng Chua. 2022. Learning to imagine: Integrating counterfactual thinking in neural discrete reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 57–69, Dublin, Ireland. Association for Computational Linguistics.

Qi Liu, Matt Kusner, and Phil Blunsom. 2021. Counterfactual data augmentation for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 187–197, Online. Association for Computational Linguistics.

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*, abs/1907.11692.

Richard Montague. 1973. *The Proper Treatment of Quantification in Ordinary English*, pages 221–242. Springer Netherlands, Dordrecht.

Armineh Nourbakhsh, Cathy Jiao, Sameena Shah, and Carolyn Rosé. 2022. Improving compositional generalization for multi-step quantitative reasoning in question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1916–1932. Association for Computational Linguistics.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, Online. Association for Computational Linguistics.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.

Le Zhang, Zichao Yang, and Diyi Yang. 2022. TreeMix: Compositional constituency-based data augmentation for natural language understanding. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5243–5258, Seattle, United States. Association for Computational Linguistics.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. 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 3277–3287, Online. Association for Computational Linguistics.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

A Results on retrieved facts

Table 9 shows the performance of FinQANet versus FinQANet+CounterComp on retrieved facts from the FinQA dataset. Similar to gold facts, CounterComp improves program accuracy, especially on multi-step output.

Model	Program accuracy			
	1 step	2 steps	3+ steps	Overall
FinQANet	64.13	57.03	20.56	58.30
+CounterComp	**67.52**	**58.87**	**22.79**	**61.18**

Table 9: Ablation results on the FinQANet model, applied to the FinQA dataset with retrieved facts.

B Training algorithm

Algorithm 1 details our pre-indexing, sampling, and training processes. Note that the algorithm is a simplified version of our implementation, e.g. it follows a basic SGD instead of a batch SGD process, and shows the process for only one epoch.

C CounterComp for operators versus operands

CounterComp intervenes on operators, whereas operands provide another possible intervention target. As mentioned in Section 2, Learning to Image (L2I) (Li et al., 2022), which focuses on counterfactual scenarios for operands, was able to outperform TAGOP by a large margin. L2I was evaluated on TAT-QA, a dataset with a limited set of possible multi-step operations, resulting in the challenge of compositional generalization being mainly focused on operands. Since we were not able to recreate the results reported in the original L2I paper[6], instead we evaluate an operand-focused approach via a metric learning method similar to CounterComp.

Given an anchor, we generate new samples using the operations laid out in the L2I paper (i.e. SWAP, ADD, MINUS, etc.), where one or more operands are perturbed in random. We apply the same perturbation in the facts. This effectively eliminates the "imagination" component but provides a baseline that is more comparable to CounterComp. These samples are used as positive examples, whereas negative examples are randomly sampled from the batch.

[6]This could be because we failed to generate a TAT-HQA dataset that was comparable to the one used in the original paper.

Algorithm 1 Training algorithm

Training data: $\{([Q||F]^{(i)}, S^{(i)})\}_{i=1}^{I}$
Parameters: λ
Model: *model*
// Create the indices for pos and neg samples
pos_index $\leftarrow \{\}$
neg_index $\leftarrow \{\}$
for $i \in \{1, \ldots, I\}$ **do**
  $O^{(i)} \leftarrow s_1^{(i)}, s_4^{(i)}, \cdots, s_{L-3}^{(i)}$
  add_to_index(pos_index, $O^{(i)}, i$)
//p is the perturbed output and l is the location of the perturbation
  for $p, l \in$ possible_perturbations($O^{(i)}$) **do**
    $j \leftarrow$ find_matching_sample(p)
    add_to_index(neg_index, $O^{(i)}, (j, l)$)
  end for
end for
// Train (single epoch, non-batch version)
for $i \in \{1, \ldots, I\}$ **do**
  for $j \in \{1, 2, \cdots, 5\}$ **do**
// Basic model loss
    $\mathcal{L}^{(i)} \leftarrow$ loss(*model.forward*($[Q||F]^{(i)}$), $S^{(i)}$)
// Pos/neg sampling
    $O^{(i)} \leftarrow s_1^{(i)}, s_4^{(i)}, \cdots, s_{L-3}^{(i)}$
    pos_sample $\leftarrow$ sample(pos_index[$O^{(i)}$] $\setminus i$)
    neg_sample, $l \leftarrow$ sample(neg_index[$O^{(i)}$])
// Find candidate intervention spans
    $\mathcal{Q}^{(i)} \leftarrow$ find_intrvntn_span(i)
    $\mathcal{Q}_{\text{pos}}^{(i)} \leftarrow$ find_intrvntn_span(pos_sample)
    $\mathcal{Q}_{\text{neg}}^{(i)} \leftarrow$ find_intrvntn_span(neg_sample)
// Calculate edit distances and loss
    $\text{NLD}_{\text{pos}}^{(i)} \leftarrow$ norm_edit_dist($\mathcal{Q}^{(i)}, \mathcal{Q}_{\text{pos}}^{(i)}$)
    $\text{NLD}_{\text{neg}}^{(i)} \leftarrow$ norm_edit_dist($\mathcal{Q}^{(i)}, \mathcal{Q}_{\text{neg}}^{(i)}$)
    $\alpha^{(i)} = 1 - |\text{NLD}_{\text{neg}}^{(i)} - \text{NLD}_{\text{pos}}^{(i)}|$
    $\text{pos\_dist}_j^{(i)} = ||\mathbf{h}_l^{(i)} - \mathbf{h}_{l,\text{pos}}^{(i)}||_2^2$
    $\text{neg\_dist}_j^{(i)} = ||\mathbf{h}_l^{(i)} - \mathbf{h}_{l,\text{neg}}^{(i)}||_2^2$
    $\mathcal{L}_{\text{triplet}_j}^{(i)} = \max\{\text{pos\_dist}_j^{(i)} - \text{neg\_dist}_j^{(i)} + \alpha^{(i)}, 0\}$
  end for
  $\mathcal{L}^{(i)} = (1-\lambda)\mathcal{L}^{(i)} + \frac{\lambda}{5}\sum_{j=1}^{5}\mathcal{L}_{\text{triplet}_j}^{(i)}$
end for
$\mathcal{L} = \frac{1}{I}\sum_{i=1}^{I}\mathcal{L}^{(i)}$
model.backward($\mathcal{L}$)

Table 10 shows the program accuracy of CounterComp versus the new method when applied to each dataset. As expected, TAT-QA is the only dataset responsive to the perturbation of operands. All other datasets suffer from an exclusive focus on operands. For HiTab and MULTIHIERTT, the operand strategy also underperforms compared to the baseline FinQANet performance (see Table 5).

Model	FinQA	TAT-QA	HiTab	MULTIHIERTT
CounterComp (operators)	**74.49**	**70.01**	**32.61**	**40.85**
CounterComp (operands)	68.98	70.80	28.88	37.67

Table 10: Program accuracy of CounterComp versus a sampling strategy focused on operands. FinQANet was used for all experiments.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☑ A2. Did you discuss any potential risks of your work?
6.3, 6.4, 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

5.1, 5.2

☑ B1. Did you cite the creators of artifacts you used?
5.1, 5.2

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
5.1, 5.2

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
5.1, 5.2

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The data is based on publicly available corporate reports.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
5.1, 5.2

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
1

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
5

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
6

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
No packages used.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.