Break, Perturb, Build: Automatic Perturbation of Reasoning Paths Through Question Decomposition

Mor Geva, Tomer Wolfson, Jonathan Berant
School of Computer Science, Tel Aviv University, Israel
Allen Institute for Artificial Intelligence
{morgeva@mail,tomerwol@mail,joberant@cs}.tau.ac.il

Abstract

Recent efforts to create challenge benchmarks that test the abilities of natural language understanding models have largely depended on human annotations. In this work, we introduce the *''Break, Perturb, Build''* (BPB) framework for automatic reasoning-oriented perturbation of question-answer pairs. BPB represents a question by decomposing it into the reasoning steps that are required to answer it, symbolically perturbs the decomposition, and then generates new question-answer pairs. We demonstrate the effectiveness of BPB by creating evaluation sets for three reading comprehension (RC) benchmarks, generating thousands of high-quality examples without human intervention. We evaluate a range of RC models on our evaluation sets, which reveals large performance gaps on generated examples compared to the original data. Moreover, symbolic perturbations enable fine-grained analysis of the strengths and limitations of models. Last, augmenting the training data with examples generated by BPB helps close the performance gaps, without any drop on the original data distribution.

1 Introduction

Evaluating natural language understanding (NLU) systems has become a fickle enterprise. While models outperform humans on standard benchmarks, they perform poorly on a multitude of distribution shifts (Jia and Liang, 2017; Naik et al., 2018; McCoy et al., 2019, *inter alia*). To expose such gaps, recent work has proposed to evaluate models on *contrast sets* (Gardner et al., 2020), or *counterfactually-augmented data* (Kaushik et al., 2020), where minimal but meaningful perturbations are applied to test examples. However, since such examples are manually written, collecting them is expensive, and procuring diverse perturbations is challenging (Joshi and He, 2021).

Recently, methods for automatic generation of contrast sets were proposed. However, current methods are restricted to shallow surface perturbations (Mille et al., 2021; Li et al., 2020), specific reasoning skills (Asai and Hajishirzi, 2020), or rely on expensive annotations (Bitton et al., 2021). Thus, automatic generation of examples that test high-level reasoning abilities of models and their robustness to fine semantic distinctions remains an open challenge.

In this work, we propose the *''Break, Perturb, Build''* (BPB) framework for automatic generation of reasoning-focused contrast sets for reading comprehension (RC). Changing the high-level semantics of questions and generating question-answer pairs automatically is challenging. First, it requires extracting the reasoning path expressed in a question, in order to manipulate it. Second, it requires the ability to generate grammatical and coherent questions. In Figure 1, for example, transforming Q, which involves *number comparison*, into Q1, which requires *subtraction*, leads to dramatic changes in surface form. Third, it requires an automatic method for computing the answer to the perturbed question.

Our insight is that perturbing question semantics is possible when modifications are applied to a *structured meaning representation*, rather than to the question itself. Specifically, we represent questions with QDMR (Wolfson et al., 2020), a representation that decomposes a question into a sequence of reasoning steps, which are written in natural language and are easy to manipulate. Relying on a structured representation lets us develop a pipeline for perturbing the reasoning path expressed in RC examples.

Our method (see Figure 1) has four steps. We (1) parse the question into its QDMR decomposition, (2) apply rule-based perturbations to the decomposition, (3) generate new questions from

Transactions of the Association for Computational Linguistics, vol. 10, pp. 111–126, 2022. https://doi.org/10.1162/tacl_a_00450
Action Editor: Preslav Nakov. Submission batch: 8/2021; Revision batch: 9/2021; Published 2/2022.

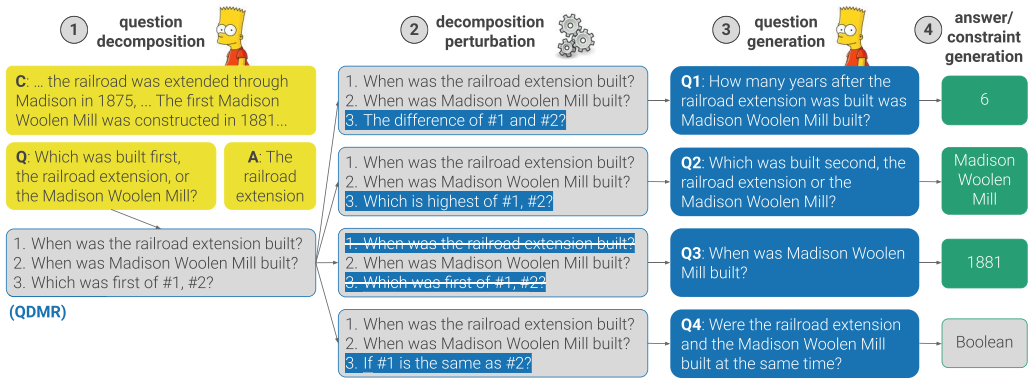

Figure 1: An overview of BPB. Given a context (C), question (Q), and the answer (A) to the question, we generate new examples by (1) parsing the question into its QDMR decomposition, (2) applying semantic perturbations to the decomposition, (3) generating a question for each transformed decomposition, and (4) computing answers/constraints to the new questions.

the perturbed decompositions, and (4) compute their answers. In cases where computing the answer is impossible, we compute constraints on the answer, which are also useful for evaluation. For example, for Q4 in Figure 1, even if we cannot extract the years of the described events, we know the answer type of the question (Boolean). Notably, aside from answer generation, all steps depend on the question only, and can be applied to other modalities, such as visual or table question answering (QA).

Running BPB on the three RC datasets, DROP (Dua et al., 2019), HotpotQA (Yang et al., 2018), and IIRC (Ferguson et al., 2020), yields thousands of semantically rich examples, covering a majority of the original examples (63.5%, 70.2%, and 45.1%, respectively). Moreover, we validate examples using crowdworkers and find that $\geq$85% of generated examples are correct.

We demonstrate the utility of BPB for comprehensive and fine-grained evaluation of multiple RC models. First, we show that leading models, such as UnifiedQA (Khashabi et al., 2020b) and TASE (Segal et al., 2020), struggle on the generated contrast sets with a decrease of 13-36 F_1 points and low consistency (<40). Moreover, analyzing model performance per perturbation type and constraints, reveals the strengths and weaknesses of models on various reasoning types. For instance, (a) models with specialized architectures are more brittle compared to general-purpose models trained on multiple datasets, (b) TASE fails to answer intermediate reasoning steps on DROP, (c) UnifiedQA fails completely on questions requiring numerical computations, and (d) models tend to do better when the numerical value of an answer is small. Last, data augmentation with examples generated by BPB closes part of the performance gap, without any decrease on the original datasets.

In summary, we introduce a novel framework for automatic perturbation of complex reasoning questions, and demonstrate its efficacy for generating contrast sets and evaluating models. We expect that imminent improvements in question generation, RC, and QDMR models will further widen the accuracy and applicability of our approach. The generated evaluation sets and codebase are publicly available at `https://github.com/mega002/qdmr-based-question-generation`.

2 Background

Our goal, given a natural language question q, is to automatically alter its semantics, generating perturbed questions $\hat{q}$ for evaluating RC models. This section provides background on the QDMR representation and the notion of *contrast sets*.

Question Decomposition Meaning Representation (QDMR). To manipulate question semantics, we rely on QDMR (Wolfson et al., 2020), a structured meaning representation for questions. The QDMR decomposition $d = \text{QDMR}(q)$ is a sequence of reasoning steps $s_1, \ldots, s_{|d|}$ required to answer q. Each step s_i in d is an intermediate question that is phrased in natural language and annotated with a logical operation o_i, such as `selection` (e.g., *''When was the Madison Woolen Mill built?''*) or `comparison` (e.g., *''Which is highest of #1, #2?''*). Example QDMRs are shown

in Figure 1 (step 2). QDMR paves a path towards controlling the reasoning path expressed in a question by changing, removing, or adding steps (§3.2).

Contrast Sets. Gardner et al. (2020) defined the contrast set $\mathcal{C}(x)$ of an example x with a label y as a set of examples with minimal perturbations to x that typically affect y. Contrast sets evaluate whether a local decision boundary around an example is captured by a model. In this work, given a question-context pair $x = \langle q, c \rangle$, we semantically perturb the question and generate examples $\hat{x} = \langle \hat{q}, c \rangle \in \mathcal{C}(\langle q, c \rangle)$ that modify the original answer a to $\hat{a}$.

3 BPB: Automatically Generating Semantic Question Perturbations

We now describe the BPB framework. Given an input $x = \langle q, c \rangle$ of question and context, and the answer a to q given c, we automatically map it to a set of new examples $\mathcal{C}(x)$ (Figure 1). Our approach uses models for question decomposition, question generation (QG), and RC.

3.1 Question Decomposition

The first step (Figure 1, step 1) is to represent q using a structured decomposition, $d = \text{QDMR}(q)$. To this end, we train a text-to-text model that generates d conditioned on q. Specifically, we fine-tune BART (Lewis et al., 2020) on the *high-level* subset of the BREAK dataset (Wolfson et al., 2020), which consists of 23.8K $\langle q, d \rangle$ pairs from three RC datasets, including DROP and HOTPOTQA.[1] Our QDMR parser obtains a 77.3 SARI score on the development set, which is near state-of-the-art on the leaderboard.[2]

3.2 Decomposition Perturbation

A decomposition d describes the reasoning steps necessary for answering q. By modifying d's steps, we can control the semantics of the question. We define a ''library'' of rules for transforming $d \rightarrow \hat{d}$, and use it to generate questions $\hat{d} \rightarrow \hat{q}$.

BPB provides a general method for creating a wide range of perturbations. In practice, though, deciding which rules to include is coupled with the reasoning abilities expected from our models. For example, there is little point in testing a model on arithmetic operations if it had never seen such examples. Thus, we implement rules based on the reasoning skills required in current RC datasets (Yang et al., 2018; Dua et al., 2019). As future benchmarks and models tackle a wider range of reasoning phenomena, one can expand the rule library.

Table 1 provides examples for all QDMR perturbations, which we describe next:

- `AppendBool`: When the question q returns a numeric value, we transform its QDMR by appending a ''yes/no'' comparison step. The comparison is against the answer a of question q. As shown in Table 1, the appended step compares the previous step result (*''#3''*) to a constant (*''is higher than 2''*). `AppendBool` perturbations are generated for 5 comparison operators ($>, <, \leq, \geq, \neq$). For the compared values, we sample from a set, based on the answer a: $\{a + k, a - k, \frac{a}{k}, a \times k\}$ for $k \in \{1, 2, 3\}$.

- `ChangeLast`: Changes the type of the last QDMR step. This perturbation is applied to steps involving operations over two referenced steps. Steps with type {`arithmetic`, `comparison`} have their type changed to either {`arithmetic`, `Boolean`}. Table 1 shows a comparison step changed to an `arithmetic` step, involving subtraction. Below it, an `arithmetic` step is changed to a yes/no question (`Boolean`).

- `ReplaceArith`: Given an `arithmetic` step, involving either subtraction or addition, we transform it by flipping its arithmetic operation.

- `ReplaceBool`: Given a `Boolean` step, verifying whether two statements are correct, we transform it to verify if neither are correct.

- `ReplaceComp`: A `comparison` step compares two values and returns the highest or lowest. Given a `comparison` step, we flip its expression from ''highest'' to ''lowest'' and vice versa.

- `PruneStep`: We remove one of the QDMR steps. Following step pruning, we prune all

[1] We fine-tune BART-large for 10 epochs, using a learning rate of $3e^{-5}$ with polynomial decay and a batch size of 32.

[2] https://leaderboard.allenai.org/break_high_level/.

Perturbation	Question	QDMR	Perturbed QDMR	Perturbed Question
`Append Boolean step`	Kadeem Jack is a player in a league that started with how many teams?	(1) league that Kadeem Jack is a player in; (2) teams that #1 started with; (3) number of #2	(1) league that Kadeem Jack is a player in; (2) teams that #1 started with; (3) number of #2; **(4) if #3 is higher than 2**	If Kadeem Jack is a player in a league that started with more than two teams?
`Change last step (to arith-metic)`	Which gallery was founded first, Hughes-Donahue Gallery or Art Euphoric?	(1) when was Hughes-Donahue Gallery founded; (2) when was Art Euphoric founded; **(3) which was first of #1, #2**	(1) when was Hughes-Donahue Gallery founded; (2) when was Art Euphoric founded; **(3) the difference of #1 and #2**	How many years after Hughes-Donahue Gallery was founded was Art Euphoric founded?
`Change last step (to Boolean)`	How many years after Madrugada's final concert did Sunday Driver become popular?	(1) year of Madrugada's final concert; (2) year when Sunday Driver become popular; **(3) the difference of #2 and #1**	(1) year of Madrugada's final concert; (2) year when Sunday Driver become popular; **(3) if #1 is the same as #2**	Did Sunday Driver become popular in the same year as Madrugada's final concert?
`Replace arith-metic op.`	How many more native Hindi speakers are there compared to native Kannada speakers?	(1) native Hindi speakers; (2) native Kannada speakers; (3) number of #1; (4) number of #2; **(5) difference of #3 and #4**	(1) native Hindi speakers; (2) native Kannada speakers; (3) number of #1; (4) number of #2; **(5) sum of #3 and #4**	Of the native Hindi speakers and native Kannada speakers, how many are there in total?
`Replace Boolean op.`	Can Stenocereus and Pachypodium both include tree like plants?	(1) if Stenocereus include tree like plants; (2) if Pachypodium include treelike plants; **(3) if both #1 and #2 are true**	(1) if Stenocereus include tree like plants; (2) if Pachypodium include treelike plants; **(3) if both #1 and #2 are false**	Do neither Stenocereus nor Pachypodium include tree like plants?
`Replace comparison op.`	Which group is smaller for the county according to the census: people or households?	(1) size of the people group in the county according to the census; (2) size of households group in the county according to the census; **(3) which is smaller of #1, #2**	(1) size of the people group in the county according to the census; (2) size of households group in the county according to the census; **(3) which is highest of #1, #2**	According to the census, which group in the county from the county is larger: people or households?
`Prune step`	How many people comprised the total adult population of Cunter, excluding seniors?	(1) adult population of Cunter; **(2) #1 excluding seniors;** (3) number of #2	(1) adult population of Cunter; (2) number of #2	How many adult population does Cunter have?

Table 1: The full list of semantic perturbations in BPB. For each perturbation, we provide an example question and its decomposition. We highlight the altered decomposition steps, along with the generated question.

other steps that are no longer referenced. We apply only a single `PruneStep` per d. Table 1 displays $\hat{d}$ after its second step has been pruned.

3.3 Question Generation

At this point (Figure 1, step 3), we parsed q to its decomposition d and altered its steps to produce the perturbed decomposition $\hat{d}$. The new $\hat{d}$ expresses a different reasoning process compared to the original q. Next, we generate the *perturbed question* $\hat{q}$ corresponding to $\hat{d}$. To this end, we train a QG model, generating questions conditioned on the input QDMR. Using the same $\langle q, d \rangle$ pairs used to train the QDMR parser (§3.1), we train a separate BART model for mapping $d \rightarrow q$.[3]

An issue with our QG model is that the perturbed $\hat{d}$ may be outside the distribution the QG

[3]We use the same hyperparameters as detailed in §3.1, except the number of epochs, which was set to 15.

Original question	Augmented question
How many interceptions **did** Matt Hasselbeck throw?	**If** Matt Hasselbeck throw **less than 23** interceptions? (`AppendBool`)
How many touchdowns **were there** in the first quarter?	**If there were two** touchdowns in the first quarter? (`AppendBool`)
Are Giuseppe Verdi **and** Ambroise Thomas **both** Opera composers?	Are **neither** Giuseppe Verdi **nor** Ambroise Thomas Opera composers? (`ReplaceBool`)
Which singer is **younger**, Shirley Manson or Jim Kerr?	Which singer is **older**, Shirley Manson or Jim Kerr? (`ReplaceComp`)

Table 2: Example application of all textual patterns used to generate questions q_{aug} (perturbation type highlighted). Boldface indicates the pattern matched in q and the modified part in q_{aug}. Decompositions d and d_{aug} omitted for brevity.

model was trained on, e.g., applying `AppendBool` on questions from DROP results in yes/no questions that do not occur in the original dataset. This can lead to low-quality questions $\hat{q}$. To improve our QG model, we use simple heuristics to take $\langle q, d\rangle$ pairs from BREAK and generate additional pairs $\langle q_{aug}, d_{aug}\rangle$. Specifically, we define 4 textual patterns, associated with the perturbations, `AppendBool`, `ReplaceBool` or `ReplaceComp`. We automatically generate examples $\langle q_{aug}, d_{aug}\rangle$ from $\langle q, d\rangle$ pairs that match a pattern. An example application of all patterns is in Table 2. For example, in `AppendBool`, the question q_{aug} is inferred with the pattern *''how many . . . did''*. In `ReplaceComp`, generating q_{aug} is done by identifying the superlative in q and fetching its antonym.

Overall, we generate 4,315 examples and train our QG model on the union of BREAK and the augmented data. As QG models have been rapidly improving, we expect future QG models will be able to generate high-quality questions for any decomposition without data augmentation.

3.4 Answer Generation

We converted the input question into a set of perturbed questions without using the answer or context. Therefore, this part of BPB can be applied to any question, regardless of the context modality. We now describe a RC-specific component for answer generation that uses the textual context.

To get complete RC examples, we must compute answers to the generated questions (Figure 1, step 4). We take a two-step approach: For some questions, we can compute the answer automatically based on the type of applied perturbation. If this fails, we compute the answer by answering each step in the perturbed QDMR $\hat{d}$.

Answer Generation Methods. Let $\langle q, c, a\rangle$ be the original RC example and denote by $\hat{q}$ the generated question. We use the following per-perturbation rules to generate the new answer $\hat{a}$:

- `AppendBool`: The transformed $\hat{q}$ compares whether the answer a and a numeric value v satisfy a comparison condition. As the values of a and v are given (§3.2), we can compute whether the answer is *''yes''* or *''no''* directly.

- `ReplaceArith`: This perturbation converts an answer that is the sum (difference) of numbers to an answer that is the difference (sum). We can often identify the numbers by looking for numbers x, y in the context c such that $a = x \pm y$ and flipping the operation: $\hat{a} = |x \mp y|$. To avoid noise, we discard examples for which there is more than one pair of numbers that result in a, and cases where $a < 10$, as the computation may involve explicit counting rather than an arithmetic computation.

- `ReplaceBool`: This perturbation turns a verification of whether two statements x, y are true, to a verification of whether neither x nor y are true. Therefore, if a is *''yes''* (i.e., both x, y are true), $\hat{a}$ must be *''no''*.

- `ReplaceComp`: This perturbation takes a comparison question q that contains two candidate answers x, y, of which x is the answer a. We parse q with spaCy[4] and identify the two answer candidates x, y, and return the one that is not a.

[4] `https://spacy.io/`.

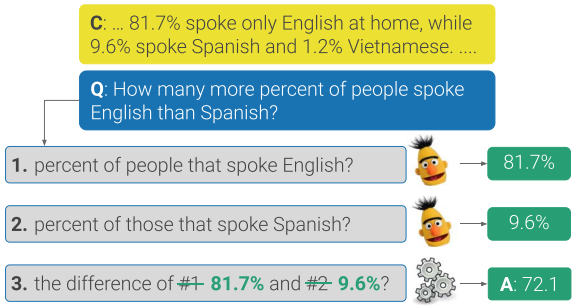

Figure 2: Example execution of the QDMR evaluator.

QDMR Evaluator. When our heuristics do not apply (e.g., arithmetic computations over more than two numbers, `PruneStep`, and `ChangeLast`), we use a RC model and the QDMR structure to directly *evaluate* each step of $\hat{d}$ and compute $\hat{a}$. Recall each QDMR step s_i is annotated with a logical operation o_i (§2). To evaluate $\hat{d}$, we go over it step-by-step, and for each step either apply the RC model for operations that require querying the context (e.g., `selection`), or directly compute the output for numerical/set-based operations (e.g., `comparison`). The answer computed for each step is then used for replacing placeholders in subsequent steps. An example is provided in Figure 2.

We discard the generated example when the RC model predicted an answer that does not match the expected argument type in a following step for which the answer is an argument (e.g., when a non-numerical span predicted by the RC model is used as an argument for an `arithmetic` operation), and when the generated answer has more than 8 words. Also, we discard operations that often produce noisy answers based on manual analysis (e.g., `project` with a non-numeric answer).

For our QDMR evaluator, we fine-tune a RoBERTa-large model with a standard span-extraction output head on SQuAD (Rajpurkar et al., 2016) and BoolQ (Clark et al., 2019). BoolQ is included to support yes/no answers.

3.5 Answer Constraint Generation

For some perturbations, even if we fail to generate an answer, it is still possible to derive constraints on the answer. Such constraints are valuable, as they indicate cases of model failure. Therefore, in addition to $\hat{a}$, we generate four types of *answer constraints*: `Numeric`, `Boolean`, $\geq$, $\leq$.

	DROP	HPQA	IIRC
development set size	9,536	7,405	1,301
# of unique generated perturbations	65,675	10,541	3,119
# of generated examples	61,231	8,488	2,450
# of covered development examples	6,053	5,199	587
% of covered development examples	63.5	70.2	45.1
Avg. contrast set size	11.1	2.6	5.2
Avg. # of perturbations per example	1.2	1	1
% of answers generated by the QDMR evaluator	5.8	61.8	22.5
# of annotated contrast examples	1,235	1,325	559
% of valid annotated examples	85	89	90.3

Table 3: Generation and annotation statistics for the DROP, HotpotQA, and IIRC datasets.

When changing the last QDMR step to an arithmetic or Boolean operation (Table 1, rows 2-3), the new answer should be `Numeric` or `Boolean`, respectively. An example for a `Boolean` constraint is given in Q4 in Figure 1. When replacing an arithmetic operation (Table 1, row 4), if an answer that is the sum (difference) of two non-negative numbers is changed to the difference (sum) of these numbers, the new answer must not be greater (smaller) than the original answer. For example, the answer to the question perturbed by `ReplaceArith` in Table 1 (row 4) should satisfy the $\geq$ constraint.

4 Generated Evaluation Sets

We run BPB on the RC datasets DROP (Dua et al., 2019), HotpotQA (Yang et al., 2018), and IIRC (Ferguson et al., 2020). Questions from the training sets of DROP and HotpotQA are included in Break, and were used to train the decomposition and QG models. Results on IIRC show BPB's generalization to datasets for which we did not observe $\langle q, d \rangle$ pairs. Statistics on the generated contrast and constraint sets are in Table 3, 4, and 5.

Contrast Sets. Table 3 shows that BPB successfully generates thousands of perturbations for each dataset. For the vast majority of perturbations, answer generation successfully produced a result—for 61K out of 65K in DROP, 8.5K out of 10.5K in HotpotQA, and 2.5K out of 3K in IIRC. Overall, 61K/8.5K examples were created

		DROP	HPQA	IIRC
	contrast	56,205	2,754	1,884
AppendBool	annotated	254	200	198
	% valid	**97.2**	**98**	**98**
	contrast	85	408	43
ChangeLast	annotated	69	200	43
	% valid	**55.1**	**84.5**	**76.7**
	contrast	390	–	1
ReplaceArith	annotated	191	–	1
	% valid	**79.6**	–	0
	contrast	–	127	1
ReplaceBool	annotated	–	127	1
	% valid	–	**97.6**	100
	contrast	1,126	362	14
ReplaceComp	annotated	245	200	14
	% valid	**90.2**	**88.5**	71.4
	contrast	3,425	3,777	507
PruneStep	annotated	476	399	302
	% valid	**82.4**	**85.8**	**88.4**

Table 4: Per-perturbation statistics for generation and annotation of our datasets. Validation results are in bold for perturbations with at least 40 examples.

from the development sets of DROP/ HotpotQA, respectively, covering 63.5%/70.2% of the development set. For the held-out dataset IIRC, not used to train the QDMR parser and QG model, BPB created a contrast set of 2.5K examples, which covers almost half of the development set.

Table 4 shows the number of generated examples per perturbation. The distribution over perturbations is skewed, with some perturbations (AppendBool) 100x more frequent than others (ReplaceArith). This is because the original distribution over operations is not uniform and each perturbation operates on different decompositions (e.g., AppendBool can be applied to any question with a numeric answer, while ReplaceComp operates on questions comparing two objects).

Constraint Sets. Table 5 shows the number of generated answer constraints for each dataset. The constraint set for DROP is the largest, consisting of 3.3K constraints, 8.9% of which covering DROP examples for which we could not generate a contrast set. This is due to the examples with arithmetic operations, for which it is easier to generate constraints. The constraint sets of HotpotQA and IIRC contain yes/no questions, for which we use the Boolean constraint.

Estimating Example Quality To analyze the quality of generated examples, we sampled

	DROP	HPQA	IIRC
# of constraints	3,323	550	56
% of constraints that cover examples without a contrast set	8.9	26	21.4
% of covered development examples	22.5	7.4	4
Numeric	2,398	–	–
Boolean	–	549	52
$\geq$	825	–	1
$\leq$	100	1	3

Table 5: Generation of constraints statistics for the DROP, HotpotQA, and IIRC datasets.

200-500 examples from each perturbation and dataset (unless fewer than 200 examples were generated) and let crowdworkers validate their correctness. We qualify 5 workers, and establish a feedback protocol where we review work and send feedback after every annotation batch (Nangia et al., 2021). Each generated example was validated by three workers, and is considered valid if approved by the majority. Overall, we observe a Fleiss Kappa (Fleiss, 1971) of 0.71, indicating substantial annotator agreement (Landis and Koch, 1977).

Results are in Table 3 and 4. The vast majority of generated examples ($\geq$85%) were marked as valid, showing that BPB produces high-quality examples. Moreover (Table 4), we see variance across perturbations, where some perturbations reach >95% valid examples (AppendBool, ReplaceBool), while others (ChangeLast) have lower validity. Thus, overall quality can be controlled by choosing specific perturbations.

Manual validation of generated contrast sets is cheaper than authoring contrast sets from scratch: The median validation time per example is 31 seconds, roughly an order of magnitude faster than reported in Gardner et al. (2020). Thus, when a very clean evaluation set is needed, BPB can dramatically reduce the cost of manual annotation.

Error Analysis of the QDMR Parser To study the impact of errors by the QDMR parser on the quality of generated examples, we (the authors) took the examples annotated by crowdworkers, and analyzed the generated QDMRs for 60 examples per perturbation from each dataset: 30 that were marked as valid by crowdworkers, and 30 that were marked as invalid. Specifically, for each example, we checked whether the generated

QDMR faithfully expresses the reasoning path required to answer the question, and compared the quality of QDMRs of valid and invalid examples.

For the examples that were marked as valid, we observed that the accuracy of QDMR structures is high: 89.5%, 92.7%, and 91.1% for DROP, HotpotQA, and IIRC, respectively. This implies that, overall, our QDMR parser generated faithful and accurate representations for the input questions. Moreover, for examples marked as invalid, the QDMR parser accuracy was lower but still relatively high, with 82.0%, 82.9%, and 75.5% valid QDMRs for DROP, HotpotQA, and IIRC, respectively. This suggests that the impact of errors made by the QDMR parser on generated examples is moderate.

5 Experimental Setting

We use the generated contrast and constraints sets to evaluate the performance of strong RC models.

5.1 Models

To evaluate our approach, we examine a suite of models that perform well on current RC benchmarks, and that are diverse it terms of their architecture and the reasoning skills they address:

- TASE (Segal et al., 2020): A RoBERTa model (Liu et al., 2019) with 4 specialized output heads for (a) tag-based multi-span extraction, (b) single-span extraction, (c) signed number combinations, and (d) counting (until 9). TASE obtains near state-of-the-art performance when fine-tuned on DROP.
- UnifiedQA (Khashabi et al., 2020b): A text-to-text T5 model (Raffel et al., 2020) that was fine-tuned on multiple QA datasets with different answer formats (yes/no, span, etc.). UnifiedQA has demonstrated high performance on a wide range of QA benchmarks.
- Reader (Asai et al., 2020): A BERT-based model (Devlin et al., 2019) for RC with two output heads for answer classification to `yes`/`no`/`span`/`no-answer`, and span extraction.

We fine-tune two TASE models, one on DROP and another on IIRC, which also requires numerical reasoning. Reader is fine-tuned on HotpotQA, while separate UnifiedQA models are fine-tuned on each of the three datasets. In addition, we evaluate UnifiedQA without fine-tuning, to analyze its generalization to unseen QA distributions. We denote by UnifiedQA the model without fine-tuning, and by $\text{UnifiedQA}_{\text{X}}$ the UnifiedQA model fine-tuned on dataset X.

We consider a ''pure'' RC setting, where only the context necessary for answering is given as input. For HotpotQA, we feed the model with the two gold paragraphs (without distractors), and for IIRC we concatenate the input paragraph with the gold evidence pieces from other paragraphs.

Overall, we study 6 model-dataset combinations, with 2 models per dataset. For each model, we perform a hyperparameter search and train 3-4 instances with different random seeds, using the best configuration on the development set.

5.2 Evaluation

We evaluate each model in multiple settings: (a) the original development set; (b) the generated contrast set, denoted by CONT; (c) the subset of CONT marked as valid by crowdworkers, denoted by CONT_{VAL}. Notably, CONT and CONT_{VAL} have a different distribution over perturbations. To account for this discrepancy, we also evaluate models on a sample from CONT, denoted by $\text{CONT}_{\text{RAND}}$, where sampling is according to the perturbation distribution in CONT_{VAL}. Last, to assess the utility of constraint sets, we enrich the contrast set of each example with its corresponding constraints, denoted by $\text{CONT}_{+\text{CONST}}$.

Performance is measured using the standard F_1 metric. In addition, we measure *consistency* (Gardner et al., 2020), that is, the fraction of examples such that the model predicted the correct answer to the original example as well as to all examples generated for this example. A prediction is considered correct if the F_1 score, with respect to the gold answer, is ≥ 0.8. Formally, for a set of evaluation examples $\mathcal{S} = \{\langle q_i, c_i, a_i \rangle\}_{i=1}^{|\mathcal{S}|}$:

$$\textit{consistency}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} g(\mathcal{C}(x))$$

$$g(\mathcal{X}) = \begin{cases} 1, & \text{if } \forall \langle \hat{x}, \hat{a} \rangle \in \mathcal{X} : \text{F}_1(y(\hat{x}), \hat{a}) \geq 0.8 \\ 0, & \text{otherwise} \end{cases}$$

	DEV F_1	CONT$_{VAL}$ F_1	CONT$_{RAND}$ F_1	CONT F_1	CONT$_{VAL}$ Cnst.	CONT Cnst.	CONT$_{+CONST}$ Cnst.
TASE$_{DROP}$	83.5 ± 0.1	65.9 ± 1	57.3 ± 0.6	54.8 ± 0.4	55.7 ± 1.1	35.7 ± 0.5	33.7 ± 0.3
TASE$_{DROP}$+	83.7 ± 1.1	75.2 ± 0.5	68 ± 1	66.5 ± 0.5	66.3 ± 0.4	48.9 ± 0.6	45 ± 0.4
TASE$_{IIRC}$	69.9 ± 0.5	45 ± 5	41.2 ± 3.8	33.7 ± 2.2	23.7 ± 4.7	24.3 ± 5.3	24.3 ± 5.3
TASE$_{IIRC}$+	68.8 ± 1.3	81.1 ± 4.6	78.2 ± 4.9	72.4 ± 5.7	50.4 ± 3.2	48.2 ± 2.5	48.2 ± 2.5

Table 6: Evaluation results of TASE on DROP and IIRC. For each dataset, we compare the model trained on the original and augmented (marked with +) training data.

	DEV F_1	CONT$_{VAL}$ F_1	CONT$_{RAND}$ F_1	CONT F_1	CONT$_{VAL}$ Cnst.	CONT Cnst.	CONT$_{+CONST}$ Cnst.
READER	82.2 ± 0.2	58.1 ± 0.1	54.5 ± 0.7	49.9 ± 0.4	39.6 ± 0.6	43.1 ± 0.1	43 ± 0.1
READER+	82.7 ± 0.9	89.1 ± 0.4	86.6 ± 0.6	81.9 ± 0.3	65.6 ± 0.4	56.4 ± 0.4	56.3 ± 0.4

Table 7: Results of READER on HOTPOTQA, when trained on the original and augmented (marked with +) data.

	DEV F_1	CONT$_{VAL}$ F_1	CONT$_{RAND}$ F_1	CONT F_1	CONT$_{VAL}$ Cnst.	CONT Cnst.	CONT$_{+CONST}$ Cnst.
UNIFIEDQA	28.2	38.1	35.1	34.9	5.3	4.4	2.2
UNIFIEDQA$_{DROP}$	33.9 ± 0.9	28.4 ± 0.8	26.9 ± 0.5	8.1 ± 3.8	12.2 ± 1.6	5.1 ± 0.7	4.4 ± 0.5
UNIFIEDQA$_{DROP}$+	32.9 ± 1.2	37.9 ± 1.4	35.9 ± 2.5	10.5 ± 4.4	16.9 ± 0.2	9.6 ± 0.2	8 ± 0.5
UNIFIEDQA	68.7	68.2	52.9	65.2	29.8	38.4	37.6
UNIFIEDQA$_{HPQA}$	74.7 ± 0.2	60.3 ± 0.8	58.7 ± 0.9	61.9 ± 0.7	35.6 ± 1.1	40.2 ± 0.1	39.9 ± 0.1
UNIFIEDQA$_{HPQA}$+	74.1 ± 0.2	60.3 ± 1.9	59.2 ± 1.5	62.3 ± 2.3	36.3 ± 0.7	41.6 ± 0.3	41.3 ± 0.4
UNIFIEDQA	44.5	61.1	57.2	36.5	21.6	28.1	28.1
UNIFIEDQA$_{IIRC}$	50.2 ± 0.7	45.1 ± 2.1	42.5 ± 2.3	20.4 ± 2.9	24.9 ± 1.2	28.6 ± 0.8	28.5 ± 0.8
UNIFIEDQA$_{IIRC}$+	51.7 ± 0.9	62.9 ± 2.9	54.5 ± 3.9	40.8 ± 5.4	30.2 ± 2.7	32.1 ± 1.9	32.1 ± 1.9

Table 8: Evaluation results of UNIFIEDQA on DROP, HOTPOTQA, and IIRC. We compare UNIFIEDQA without fine-tuning, and after fine-tuning on the original training data and on the augmented training data (marked with +).

where $\mathcal{C}(x)$ is the generated contrast set for example x (which includes x),[5] and $y(\hat{x})$ is the model's prediction for example $\hat{x}$. Constraint satisfaction is measured using a binary 0-1 score.

Because yes/no questions do not exist in DROP, we do not evaluate TASE$_{DROP}$ on `AppendBool` examples, which have yes/no answers, as we cannot expect the model to answer those correctly.

5.3 Results

Results are presented separately for each model, in Table 6, 7, and 8. Comparing performance on the development sets (DEV F_1) to the corresponding contrast sets (CONT F_1), we see a substantial decrease in performance on the generated contrast sets, across all datasets (e.g., 83.5 $\rightarrow$ 54.8 for TASE$_{DROP}$, 82.2 $\rightarrow$ 49.9 for READER, and 50.2 $\rightarrow$ 20.4 for UNIFIEDQA$_{IIRC}$). Moreover, model consistency (CONT Cnst.) is considerably lower than the development scores (DEV F_1), for example, TASE$_{IIRC}$ obtains 69.9 F_1 score but only 24.3 consistency. This suggests that, overall, the models do not generalize to pertrubations in the reasoning path expressed in the original question.

Comparing the results on the contrast sets and their validated subsets (CONT vs. CONT$_{VAL}$), performance on CONT$_{VAL}$ is better than on CONT (e.g., 58.1 versus 49.9 for READER). These gaps are due to (a) the distribution mismatch between the two sets, and (b) bad example generation. To isolate the effect of bad example generation, we can compare CONT$_{VAL}$ to CONT$_{RAND}$, which have the same distribution over perturbations, but CONT$_{RAND}$ is not validated by humans. We see that the performance of CONT$_{VAL}$ is typically $\leq$10% higher than CONT$_{RAND}$ (e.g., 58.1 vs. 54.5 for READER). Given that performance on the original

[5] With a slight abuse of notation, we overload the definition of $\mathcal{C}(x)$ from §2, such that members of $\mathcal{C}(x)$ include not just the queston and context, but also an answer.

development set is dramatically higher, it seems we can currently use automatically generated contrast sets (without verification) to evaluate robustness to reasoning perturbations.

Last, adding constraints to the generated contrast sets (CONT vs. $\text{CONT}_{+\text{CONST}}$) often leads to a decrease in model consistency, most notably on DROP, where there are arithmetic constraints and not only answer type constraints.

For instance, consistency drops from 35.7 to 33.7 for TASE, and from 5.1 to 4.4 for $\text{UnifiedQA}_{\text{DROP}}$. This shows that the generated constraints expose additional flaws in current models.

5.4 Data Augmentation

Results in §5.3 reveal clear performance gaps in current QA models. A natural solution is to augment the training data with examples from the contrast set distribution, which can be done effortlessly, since BPB is fully automatic.

We run BPB on the training sets of DROP, HotpotQA, and IIRC. As BPB generates many examples, it can shift the original training distribution dramatically. Thus, we limit the number of examples generated by each perturbation by a threshold τ. Specifically, for a training set $\mathcal{S}$ with $|\mathcal{S}| = n$ examples, we augment $\mathcal{S}$ with $\tau * n$ randomly generated examples from each perturbation (if fewer than $\tau * n$ examples were generated we add all of them). We experiment with three values $\tau \in \{0.03, 0.05, 0.1\}$, and choose the trained model with the best F_1 on the contrast set.

Augmentation results are shown in Table 6–8. Consistency (CONT and CONT_{VAL}) improves dramatically, with only a small change in the model's DEV performance, across all models. We observe an increase in consistency of 13 points for $\text{TASE}_{\text{DROP}}$, 24 for $\text{TASE}_{\text{IIRC}}$, 13 for Reader, and 1-4 points for the UnifiedQA models. Interestingly, augmentation is less helpful for UnifiedQA than for TASE and Reader. We conjecture that this is because UnifiedQA was trained on examples from multiple QA datasets and is thus less affected by the augmented data.

Improvement on test examples sampled from the augmented training distribution is expected. To test whether augmented data improves robustness on other distributions, we evaluate TASE+ and $\text{UnifiedQA}_{\text{DROP}}$+ on the DROP contrast set manually collected by Gardner et al. (2020). We find that training on the augmented training set does not lead to a significant change on the manually collected contrast set ($F_1$of 60.4 $\rightarrow$ 61.1 for TASE, and 30 $\rightarrow$ 29.6 for $\text{UnifiedQA}_{\text{DROP}}$). This agrees with findings that data augmentation with respect to a phenomenon may not improve generalization to other out-of-distribution examples (Kaushik et al., 2021; Joshi and He, 2021).

6 Performance Analysis

Analysis Across Perturbations. We compare model performance on the original (ORIG) and generated examples (CONT and CONT_{VAL}) across perturbations (Figure 3, 4, 5). Starting from models with specialized architectures (TASE and Reader), except for `ChangeLast` (discussed later), models' performance decreases on all perturbations. Specifically, TASE (Figure 3, 5) demonstrates brittleness to changes in comparison questions (10-30 F_1 decrease on `ReplaceComp`) and arithmetic computations ($\sim$30 F_1 decrease on `ReplaceArith`). The biggest decrease of almost 50 points is on examples generated by `PruneStep` from DROP (Figure 3), showing that the model struggles to answer intermediate reasoning steps.

Reader (Figure 4) shows similar trends to TASE, with a dramatic performance decrease of 80-90 points on yes/no questions created by `AppendBool` and `ReplaceBool`. Interestingly, Reader obtains high performance on `PruneStep` examples, as opposed to $\text{TASE}_{\text{DROP}}$ (Figure 3), which has a similar span extraction head that is required for these examples. This is possibly due to the ''train-easy'' subset of HotpotQA, which includes single-step selection questions.

Moving to the general-purpose UnifiedQA models, they perform on `PruneStep` at least as well the original examples, showing their ability to answer simple selection questions. They also demonstrate robustness on `ReplaceBool`. Yet, they struggle on numeric comparison questions or arithmetic calculations: $\sim$65 points decrease on `ChangeLast` on DROP (Figure 3), 10-30 F_1 decrease on `ReplaceComp` and `AppendBool` (Figure 3, 4, 5), and almost 0 F_1 on `ReplaceArith` (Figure 3).

Performance on CONT and CONT_{VAL}. Results on CONT_{VAL} are generally higher than CONT due to the noise in example generation. However,

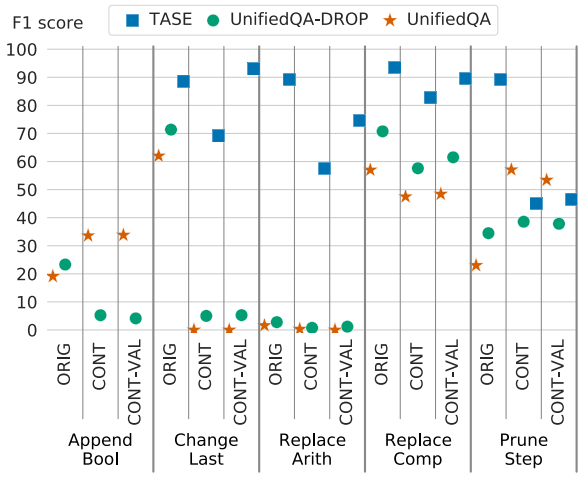

Figure 3: Performance on DROP per perturbation: on the generated contrast set (CONT), on the examples from which CONT was generated (ORIG), and on the validated subset of CONT (CONT_{VAL}).

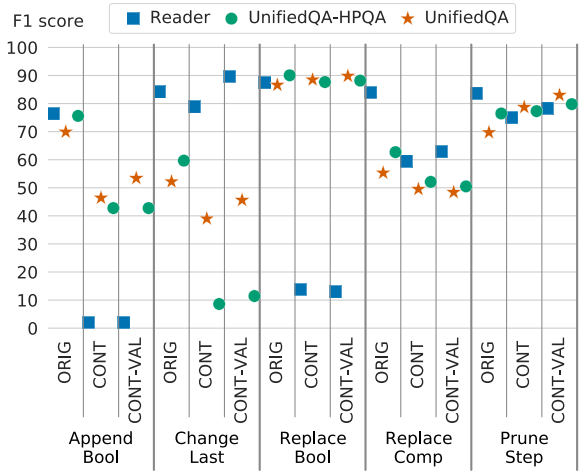

Figure 4: Performance on HotpotQA per perturbation: on the generated contrast set (CONT), on the examples from which CONT was generated (ORIG), and the validated subset of CONT (CONT_{VAL}).

whenever results on ORIG are higher than CONT, they are also higher than CONT_{VAL}, showing that the general trend can be inferred from CONT, due to the large performance gap between ORIG and CONT. An exception is `ChangeLast` in DROP and HotpotQA, where performance on CONT is lower than ORIG, but on CONT_{VAL} is higher. This is probably due to the noise in generation, especially for DROP, where example validity is at 55.1% (see Table 4).

Evaluation on Answer Constraints Evaluating whether the model satisfies answer constraints can help assess the model's skills. To this end, we

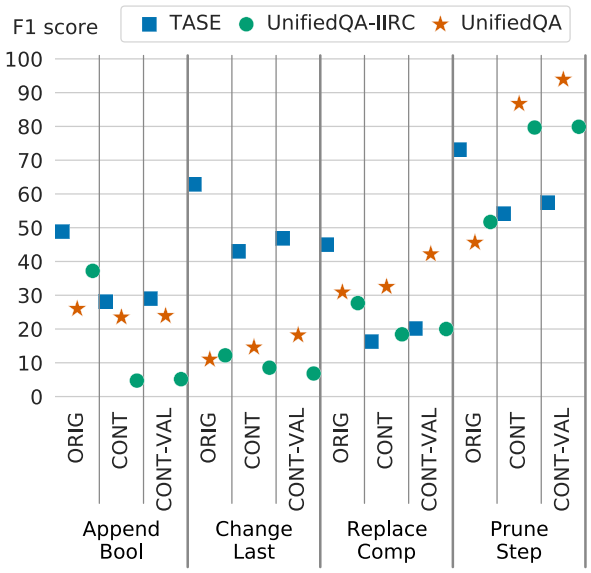

Figure 5: Performance on IIRC per perturbation: on the generated contrast set (CONT), on the examples from which CONT was generated (ORIG), and the validated subset of CONT (CONT_{VAL}).

measure the fraction of answer constraints satisfied by the predictions of each model (we consider only constraints with more than 50 examples).

Models typically predict the correct answer type; $\text{TASE}_{\text{DROP}}$ and UnifiedQA predict a number for $\geq$ 86% of the generated numeric questions, and Reader and $\text{TASE}_{\text{IIRC}}$ successfully predict a yes/no answer in $\geq$ 92% of the cases. However, fine-tuning UnifiedQA on HotpotQA and IIRC reduces constraint satisfaction (94.7 $\rightarrow$ 76.3 for $\text{UnifiedQA}_{\text{HPQA}}$, 65.4 $\rightarrow$ 38.9 for $\text{UnifiedQA}_{\text{IIRC}}$), possibly since yes/no questions constitute fewer than 10% of the examples (Yang et al., 2018; Ferguson et al., 2020). In addition, results on DROP for the constraint '$\geq$' are considerably lower than for '$\leq$' for UnifiedQA (83 $\rightarrow$ 67.4) and $\text{UnifiedQA}_{\text{DROP}}$ (81.8 $\rightarrow$ 65.9), indicating a bias towards predicting small numbers.

7 Related Work

The evaluation crisis in NLU has led to wide interest in challenge sets that evaluate the robustness of models to input perturbations. However, most past approaches (Ribeiro et al., 2020; Gardner et al., 2020; Khashabi et al., 2020a; Kaushik et al., 2020) involve a human-in-the-loop and are thus costly.

Recently, more and more work has considered using meaning representations of language to

automatically generate evaluation sets. Past work used an ERG grammar (Li et al., 2020) and AMR (Rakshit and Flanigan, 2021) to generate relatively shallow perturbations. In parallel to this work, Ross et al. (2021) used control codes over SRL to generate more semantic perturbations to declarative sentences. We generate perturbations at the level of the *underlying reasoning process*, in the context of QA. Last, Bitton et al. (2021) used scene graphs to generate examples for visual QA. However, they assumed the existence of gold scene graph at the input. Overall, this body of work represents an exciting new research program, where structured representations are leveraged to test and improve the blind spots of pre-trained language models.

More broadly, interest in automatic creation of evaluation sets that test out-of-distribution generalization has skyrocketed, whether using heuristics (Asai and Hajishirzi, 2020; Wu et al., 2021), data splits (Finegan-Dollak et al., 2018; Keysers et al., 2020), adversarial methods (Alzantot et al., 2018), or an aggregation of the above (Mille et al., 2021; Goel et al., 2021).

Last, QDMR-to-question generation is broadly related to work on text generation from structured data (Nan et al., 2021; Novikova et al., 2017; Shu et al., 2021), and to passage-to-question generation methods (Du et al., 2017; Wang et al., 2020; Duan et al., 2017) that, in contrast to our work, focused on simple questions not requiring reasoning.

8 Discussion

We propose the BPB framework for generating high-quality reasoning-focused question perturbations, and demonstrate its utility for constructing contrast sets and evaluating RC models.

While we focus on RC, our method for perturbing questions is independent of the context modality. Thus, porting our approach to other modalities only requires a method for computing the answer to perturbed questions. Moreover, BPB provides a general-purpose mechanism for question generation, which can be used outside QA as well.

We provide a library of perturbations that is a function of the current abilities of RC models. As future RC models, QDMR parsers, and QG models improve, we can expand this library to support additional semantic phenomena.

Last, we showed that constraint sets are useful for evaluation. Future work can use constraints as a supervision signal, similar to Dua et al. (2021), who leveraged dependencies between training examples to enhance model performance.

Limitations BPB represents questions with QDMR, which is geared towards representing complex factoid questions that involve multiple reasoning steps. Thus, BPB cannot be used when questions involve a single step, for example, one cannot use BPB to perturb *''Where was Barack Obama born?''*. Inherently, the effectiveness of our pipeline approach depends on the performance of its modules—the QDMR parser, the QG model, and the single-hop RC model used for QDMR evaluation. However, our results suggest that current models already yield high-quality examples, and model performance is expected to improve over time.

Acknowledgments

We thank Yuxiang Wu, Itay Levy, and Inbar Oren for the helpful feedback and suggestions. This research was supported in part by The Yandex Initiative for Machine Learning, and The European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800). This work was completed in partial fulfillment for the Ph.D. degree of Mor Geva.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1316

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.499

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations*.

Yonatan Bitton, Gabriel Stanovsky, Roy Schwartz, and Michael Elhadad. 2021. Automatic generation of contrast sets from scene graphs: Probing the compositional consistency of GQA. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 94–105, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.9

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*, pages 4171–4186, Minneapolis, Minnesota.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Dheeru Dua, Pradeep Dasigi, Sameer Singh, and Matt Gardner. 2021. Learning with instance bundles for reading comprehension. *arXiv preprint arXiv:2104.08735*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1090

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1137–1147, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.86

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1033

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378. https://doi.org/10.1037/h0031619

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.117

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 42–55, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-demos.6

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/D17-1215

Nitish Joshi and He He. 2021. An investigation of the (in) effectiveness of counterfactually augmented data. *arXiv preprint arXiv:2107.00753*.

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*.

Divyansh Kaushik, Douwe Kiela, Zachary C. Lipton, and Wen-tau Yih. 2021. On the efficacy of adversarial data collection for question answering: Results from a large-scale randomized study. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. https://doi.org/10.18653/v1/2021.acl-long.517

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Daniel Khashabi, Tushar Khot, and Ashish Sabharwal. 2020a. More bang for your buck: Natural perturbation for robust question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 163–170, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.12

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020b. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.171

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174. https://doi.org/10.2307/2529310, PubMed: 843571

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Chuanrong Li, Lin Shengshuo, Zeyu Liu, Xinyi Wu, Xuhui Zhou, and Shane Steinert-Threlkeld. 2020. Linguistically-informed transformations (LIT): A method for automatically generating contrast sets. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 126–135, 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 preprint arXiv:1907.11692*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference.

In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1334

Simon Mille, Kaustubh Dhole, Saad Mahamood, Laura Perez-Beltrachini, Varun Gangal, Mihir Kale, Emiel van Miltenburg, and Sebastian Gehrmann. 2021. Automatic construction of evaluation suites for natural language generation datasets. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Nikita Nangia, Saku Sugawara, Harsh Trivedi, Alex Warstadt, Clara Vania, and Samuel R. Bowman. 2021. What ingredients make for an effective crowdsourcing protocol for difficult NLU data collection tasks? 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 1221–1235, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.98

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-5525

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Geetanjali Rakshit and Jeffrey Flanigan. 2021. ASQ: Automatically generating question-answer pairs using AMRS. *arXiv preprint arXiv:2105.10023*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.442

Alexis Ross, Tongshuang Wu, Hao Peng, Matthew E. Peters, and Matt Gardner. 2021. Tailor: Generating and perturbing text with semantic controls. *arXiv preprint arXiv:2107.07150*.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. A simple and effective model for answering multi-span questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.248

Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. 2021.

Logic-consistency text generation from semantic parses. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4414–4426, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.388

Siyuan Wang, Zhongyu Wei, Zhihao Fan, Zengfeng Huang, Weijian Sun, Qi Zhang, and Xuanjing Huang. 2020. PathQG: Neural question generation from facts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9066–9075, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.729

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. BREAK it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics (TACL)*, 8:183–198. https://doi.org/10.1162/tacl_a_00309

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. 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 6707–6723, Online. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/D18-1259