Measure	SAM	NFL
positive correlation ↑		
m_1: Adj. sentence w2v similarity	0.58	0.67
negative correlation ↓		
m_2: Type-token ratio	0.72	0.66
m_3: Adj. sentence verb overlap	0.17	0.24
m_4: Pronoun-noun-ratio	0.07	0.05

Table 3: Detailed breakdown of measures used to obtain the final *Naturality* metric for the evaluation of the generated data.

SM1: Additional Details on the Challenge Set Generation

Algorithm 1 describes the challenge set generation in pseudo-code. The data used for generation were obtained by scraping football match reports from news and Wikipedia world cup finals websites[2][3]. They were automatically processed with the AllenNLP constituency parser [4] and manually arranged by their semantic content to form the generative grammar. Sentences were processed by the AllenNLP NER [5] and SRL [6] tools to substitute semantic roles of interest (e.g. player names, timestamps, verbs describing relevant actions) with variables, the output was manually verified and curated, resulting in the seed templates.

A detailed breakdown of the measures used to evaluate the quality of generated data is shown in Table 3. We measure the global and sentence-level indices that were reported to correlate with human judgements of writing quality by Crossley, Kyle, and McNamara (2016) and Crossley, Kyle, and Dascalu (2019). We use the associated tool TAACO to establish these measures. For the first three measures reported in Table 3 multiple indices are measured. For these measures we take the average of all available indices. We define the final *Naturality* metric as a combination of these measures. For simplicity we use the average:

$$Naturality = \frac{m_1 + (1 - m_2) + (1 - m_3) + (1 - m_4)}{4}$$

Note that we do not include intra-paragraph level measures, despite the fact that they are reported to correlate better with quality judgements. The reason for this is that both our generated passages and the reference DROP NFL data consist of a single paragraph.

Finally, Table 4 shows the effect of randomness on the metrics discussed in the paper. We measure the average result of 5 runs and report the standard deviation. As can be seen, for the data metrics of *Lexical Similarity* and *Naturality* as well as for the rEM_5 score, the impact of randomness is negligible. For the $DICE$ score the effect is more noticeable the lower the score, as discussed in the limitations section of the Section *Domain Independent Consistency Evaluation*.

[2]https://www.theguardian.com/tone/matchreports

[3]e.g. https://en.wikipedia.org/wiki/2006_FIFA_World_Cup_Final

[4]https://demo.allennlp.org/constituency-parsing

[5]https://demo.allennlp.org/named-entity-recognition

[6]https://demo.allennlp.org/semantic-role-labeling

Algorithm 1: generate

```
Data: question types $T$, question type event
      constraints $C$, number of examples per
      question type $N$, max. number of SAM per
      example $S$, number of events per report $n$,
      Question templates $\mathcal{T}_Q$ seed templates $\mathcal{T}_S$,
      grammar $\mathcal{G}$
$\mathcal{B}, \mathcal{C}, \mathcal{I} \leftarrow \{\}, \{\}, \{\}$
foreach $s \in 1 \dots S$ do
    foreach $i \in 1 \dots |T|$ do
        plans $\leftarrow$ generate all possible event plans for
          $T_i$ with $n$ events and $s$ modifications s.t they
          satisfy $C_i$
        plans $\leftarrow$ sample $N_i$ w/o replacement from
          plans
        reports $\leftarrow$ generate structured reports from
          each plan $\in$ plans
        permutations $\leftarrow \{\}$
        foreach $r \in$ reports do
            permutation $\leftarrow$ choose permutation of $n$
              from $\mathcal{T}_S$ respecting order of events of $r$
              and not in permutations
            add permutation to permutations
            $P \leftarrow \epsilon$
            $P' \leftarrow \epsilon$
            foreach template $t \in$ permutation do
                foreach symbol $v \in t$ do
                    l = realise $v$ using $\mathcal{G}$ with $v$ as
                      start symbol
                    append l to $P'$
                    if $v$ is not SAM then
                        append l to $P$
                    end
                end
            end
            $P'' \leftarrow$ copy $P'$ and remove modified
              sentences
            $Q, A, A' \leftarrow$ realise question and answers
              given $P, P'$ and $r$
            add $(Q, A, P)$ to $\mathcal{B}$, $(Q, A', P')$ to $\mathcal{I}$ and
              $(Q, A', P'')$ to $\mathcal{C}$
        end
    end
end
return $\mathcal{B}, \mathcal{C}, \mathcal{I}$
```

SM2: Data preparation and training details

For HOTPOTQA we concatenate multiple passages and their titles and prepend them with the [text] and [title] tokens, respectively. We further prepend the input with yes and no tokens, as some examples require this as answer. Following Devlin et al. (2019), we represent the question and context as a single sequence instance, separated by the

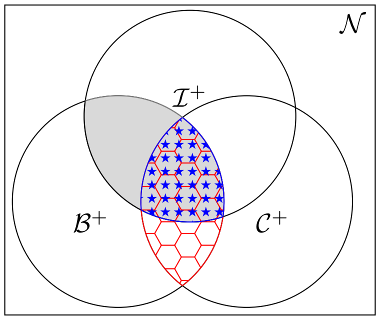

Figure 3: With circles $\mathcal{B}^+$,$\mathcal{I}^+$ and $\mathcal{C}^+$ from Eq. 1 representing instances that were answered correctly in their baseline, intervention respectively control version, respectively, $DICE$ is the proportion of the star covered area to the area covered by hexagons. Consistency, when defining contrastive sets as $\{B_i, I_i\}_{i\in\{1\ldots N\}}$ (Gardner et al. 2020) is the proportion of the grey area to the area of the entire square.

`[SEP]` token. The maximal size of this input sequence is 384 (subword) tokens, passages exceeding this length are split in multiple sequences each prepended by the question. The stride (overlap between subsequent splits of a passage) is 128 tokens. Sequences shorter than 384 are padded to maximal length. The (softmax over the) task specific layer outputs the probability distributions of tokens being the start or end index, respectively. The training objective is to minimise the cross-entropy loss between the logits of the final layer and the correct start and end indices. During inference we select the start and end index pair (s, e) with the maximum score $s + e$ with $s > e$, $e - s <=$ `max_answer_length` and neither s nor e being indices of the `SEP` or `PAD` tokens. In case the input was split, we select the pair with the highest score across all corresponding inputs.

For the generative `t5` encoder-decoder model we use a similar approach. We concatenate the question and context into a single sequence of maximal length of 512 tokens for SQuAD and DROP', 1024 for NewsQA' and 2048 for HotpotQA. We use the encoder to encode this sequence and use its hidden state as the initial representation for the decoder to generate a sequence of tokens as the answer. The training objective is to minimise the cross-entropy loss between the predicted tokens and the vocabulary indices of the expected answer. Similarly, during inference we iteratively generate a sequence of a maximum of `max_answer_length` using the hidden state of the encoder after encoding the question and passage for the first token and the hidden state of the decoder thereafter.

We implement the training and inference in PyTorch 1.6.0[7]. We use the pre-trained language models available in the `transformers`[8] library. We train the `bert`, `roberta` and `albert` models on 4 Nvidia V100 GPUs with 16 GB of RAM using data parallelism for the training on SQuAD and distributed training for the other datasets.

The `t5` models were trained using a single Nvidia V100 GPU, except when training the `t5-large` model, we employed 4-way Model Parallelism (i.e. spreading different parameters across multiple GPUs) for HotpotQA and 2-way model parallelism for `NewsQA'`, because of GPU memory constraints.

We fix the random seed to maintain deterministic behaviour and the hyper-parameters used for training are

- **Batch size:** employing (distributed) data parallelism, mixed precision and gradient accumulation we use a batch-size of 2^{10}. Note that due to this combination, the reported results on the development sets are slightly lower than what is reported in the literature (e.g. up to 3 points lower F1 score for `bert-base` and less than 1 point lower F1 score for `albert-xxlarge`). Given the training speed-up we obtain and the somewhat orthogonal goal of our study we deem this performance loss manageable.
- **Learning Rate:** We set learning rate to 5^{-5}, as it was reported to work best for the transformer training. For `t5` we found the learning rate of 0.001 used in the original experiments to work best. In both cases, we found that linearly decaying the learning rate to 0 over the course of the training is beneficial. We use the ADAM optimiser with the default parameters $\epsilon = 1 \times 10^{-8}, \beta_1 = 0.99$ and $\beta_2 = 0.999$.
- **Train Epochs:** We train on SQuAD for 3 training epochs, for 2 epochs on HotpotQA for 4 epochs on NewsQA' and for 12 epochs on DROP'. This is to ensure that the models across the different datasets have a roughly equal computational budget as the datasets vary in size and context length.
- **Maximal answer length**: We use `max_answer_length=30` when obtaining predictions on the original datasets and `max_answer_length=10` for predictions on the challenge set, because the challenge set answers are generally shorter.

The BiDAF model was trained using the AllenNLP framework using their released configuration file[9].

To replicate our experiments, please follow the "readme" in the supplemented source code.

Metric	**Mean**	**Std. dev.**
Diversity	0.1642	0
Naturality	0.66	0.003
rEM_5 on $\mathcal{B}$ ($\lvert\mathcal{B}\rvert = 4200$) of `bert-base` optimised on SQuAD	0.19	0.006
$DICE$ score of `bert-base` optimised on SQuAD	0.16	0.023

Table 4: Various measures used in the paper averaged over 5 runs where the challenge set was generated from different random seeds. Mean and standard deviation are reported.

[7]https://pytorch.org
[8]https://github.com/huggingface/transformers
[9]https://raw.githubusercontent.com/allenai/allennlp-models/v1.0.0/training_config/rc/bidaf.jsonnet

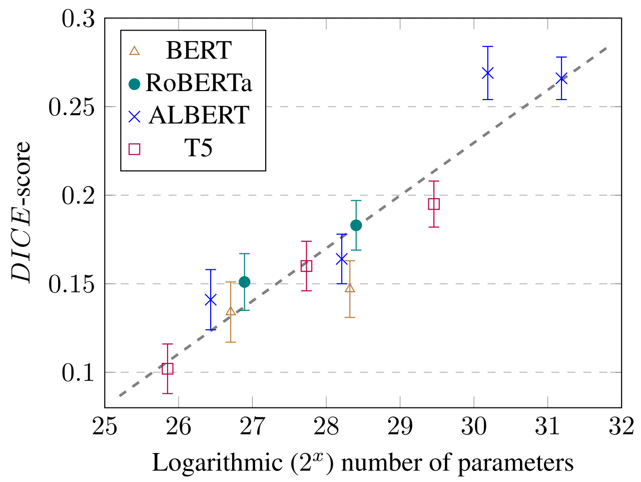

Figure 4: $DICE$ score averaged for all datasets by effective model size, shared parameters are counted separetely. The dashed line represents a fitted linear regression showing the correlation between the (logarithmic) model size and the score.

SM3: Additional Experiments and Results

We evaluate the impact of k in the rEM_k formulation and report the results in Table 6. As the parameter seems to have little impact on the final score, we settle for $k = 5$ for all relevant experiments in the paper.

We experiment with training a `bert-base` model on baseline examples without SAM and evaluate it with $DICE$ on $\mathcal{B}, \mathcal{I}, \mathcal{C}$, respectively. We use the same generated data that was used for other baselines, with 12000 baseline training instances and 2400 baseline, intervention and control evaluation examples generated from two distinct template sets. We obtain a $DICE$ score of 2 ± 1 as a result. Intuitively, this is expected and similar to what Kaushik, Hovy, and Lipton (2020) report: as the model only encounters non-altered examples during training, it is not incentivised to learn to process SAM correctly, which is required to obtain high $DICE$ scores. We take it as evidence towards the fact that the $DICE$ formulation effectively evaluates SAM performance, because models optimised on in-domain data that verifiably lacks SAM examples still obtain low $DICE$ scores.

We visualise the correlation between the effective size of the transformer language model and the $DICE$ score in Figure 4. A detailed breakdown of performance by SAM category is shown in Table 5.

SAM Category	Average $DICE$
Modal negation	20 ± 1.3
Adverbial Modification	14 ± 1.1
Implicit Negation	20 ± 1.4
Explicit Negation	26 ± 1.4
Polarity Reversing	18 ± 1.3
Negated Polarity Preserving	23 ± 1.5

Table 5: Average performance on the challenge set, by SAM category.

k	rEM_k	k	rEM_k	k	rEM_k
1	19.1 ± 1	2	19.2 ± 1	3	19.5 ± 1
4	19.5 ± 1	5	19.5 ± 1		

Table 6: Impact of k in the formulation of rEM_k when evaluating a SQuAD-optimised `bert-base` model on $\mathcal{B}$.

Label	Occurences	in %
Nonaltering	522	80.68%
Altering	1	0.15 %
QuestionAnswer	61	9.43 %
BadExample	36	5.56 %
Opposite	27	4.17 %

Table 7: Detailed results of the annotation task.

For the manual annotation of training examples from the four datasets used in the study, we sampled 100 passages from each dataset where the expressions "almost", "nearly" and "all but" from the category *I2: Adverbial Modification* were found in the passage within 100 characters of the expected answer. Because the datasets (except HotpotQA) feature more than one question per passage, the overall number of questions for annotation is 647. The examples were annotated by the first author with the following labels:

- *Nonaltering* if removing the matched expression does not change the expected answer
- *Altering* if removing the matched expression *does* change the expected answer
- *QuestionAnswer* if the matched expression is part of the question or expected answer
- *BadExample* if the match was erroneous (e.g. if "all but" was used in the sense "every single one except" rather than "almost")
- *Opposite* if the expected answer ignores the expression (e.g. when for the Question "What fraction did [...]" and the answer sentence "Nearly half of [...]" the expected answer is "half" rather than "nearly half".

20% of the passages were co-annotated by the last author, the inter-annotator agreement as per Cohen's κ is 0.82. The disagreements concerned the category *BadExample* and *Nonaltering*, with some of the labels being assigned differently by the two annotators. Besides these two categories the agreement score is in fact 1.0.

The human performance on SAM was established by means of crowd-sourcing a sample of 100 examples taken from the intervention set $\mathcal{I}$ used to evaluate the learned baselines. The annotators were asked to select a span that represents the correct answer given a question and a paragraph, without priming them on the nature of the task, (i.e. the fact that they contain SAM). Specifically, the annotators were presented the following instruction: *You will be presented a football match report and a question concerning* *said report. Read the paragraph and question* carefully *and then highlight the passage that in your opinion answers the question. All questions are answerable unambiguously by a single continuous span that is present in text. Background knowledge is not required (such as the distance of a penalty kick in football). When annotating, please annotate the full name (e.g. What Zit Tooya) and any numbers together with the belonging units (e.g. 1 pint), where appropriate. If you think an example is unanswerable, please flag it as an error and let me know.* This instruction was followed by an example annotation and a technical explanation of the annotation tool.

In order to better represent human performance rather than collect high-quality annotations, we collected a single annotation per example.

A manual analysis of erroneous annotations revealed that in 38% of the cases, SAM were ignored. The remaining errors (62%) were due to other reasons, such as confusing the interrogative pronoun (e.g. by answering with a span denoting a distance to a "When" question), or selecting otherwise wrong answers that are not related to presence of absence of SAM.