

---

**Question:** Our King has always been a little different. He’s very quiet I haven’t really seen him give anyone orders at all either. Whenever something needs to be done I see him doing it. [ . . . ] That’s when I believed everything he told me - while he took me adventuring with him. Now I know that there is a better life to be had out of this one. A less edged and more rounded one. Thank you king Steve. Minecraft Steve.

Summarize the story’s setting.

**Answer:** The story is set in a medieval kingdom, where the king is the main character of the story. The king is described as a good and active king. The narrator is a child who has always wondered why the king is so different from everyone else.

**Critiqueable:** Yes

**Critique:** The answer should mention that the setting is a video game.

**Helpful:** Yes

**Conditional refinement:** The story is set in a medieval kingdom in the video game Minecraft. The king is the main character of the story. The king is described as a good and active king. The narrator is a child who has always wondered why the king is so different from everyone else.

Table 2: Representative example of a topic-based summarization task, and its associated tasks. See Appendix A.5 for details on how we format our tasks (different than shown).

## 2.2 Topic-based summarization

We report most of our main results on the base task of topic-based summarization [Dan05, ZYY<sup>+</sup>21], a task similar to or interchangeable with query-based summarization and question-focused summarization. In topic-based summarization, the summary focuses on a specific aspect of a text rather than trying to summarize the whole text. See Table 2 for an example.

We collected our own dataset of over 6,000 distinct topical queries and summaries, on over 2,000 distinct passages. Our distribution of passages is sampled from a dataset of short stories, Wikipedia articles, or web articles (mostly news) scraped from the internet. Most tasks were generated based on short texts with less than 2,048 tokens when encoded with the GPT-2 tokenizer [RWC<sup>+</sup>19]. We also gathered some tasks based on texts with up to 4,096 tokens which were not used for training.

Our labelers generated between 1 and 8 topic-based summarization questions per passage, typically also including a topic not covered by the passage (for which the answer is empty). Summaries are up to a paragraph long – we targeted between 2-10 sentences unless the topic was missing. We aimed for these topics to be non-trivial to summarize in various ways. See Appendix A for details.

### 2.2.1 Data collection

We collect demonstrations on all the tasks mentioned in Section 2.1. Given a task for which we want to collect a demonstration, we can choose whether each input is generated from a model or human. We always use a human-generated question. All tasks but the base task require an answer as input, many for which we typically use outputs from our best model. For example, critique demonstrations are on model-generated answers, and helpfulness judgements are on model-generated critiques. For refinements the situation is more complex, and detailed in Appendix A.2.

Since we need model outputs for most demonstrations, we collect data in rounds. After each round, we train a model jointly on all task demonstrations collected thus far. We start with base task demonstration collection. Then with a model trained on only the base task, we collect demonstrations for critiqueability, critique, and refinement tasks using model-generated answers. Finally, we collect demonstrations for helpfulness tasks, by showing labelers model-generated critiques of model-generated answers.

For more details on our data collection, see Appendix A and Table 4. We publicly release all data used to train final models<sup>2</sup>.

---

<sup>2</sup>We release six files, located at <https://openaipublic.blob.core.windows.net/critiques/dataset/:base/train.jsonl.gz>, [base/test.jsonl.gz](https://openaipublic.blob.core.windows.net/critiques/dataset/:base/test.jsonl.gz), [critiques/train.jsonl.gz](https://openaipublic.blob.core.windows.net/critiques/dataset/:critiques/train.jsonl.gz), [critiques/test.jsonl.gz](https://openaipublic.blob.core.windows.net/critiques/dataset/:critiques/test.jsonl.gz), [helpfulness/train.jsonl.gz](https://openaipublic.blob.core.windows.net/critiques/dataset/:helpfulness/train.jsonl.gz), [helpfulness/test.jsonl.gz](https://openaipublic.blob.core.windows.net/critiques/dataset/:helpfulness/test.jsonl.gz)

### 2.2.2 Models

Similarly to [DL15, RNSS18, BHA<sup>+</sup>21], we start with foundation models pre-trained to autoregressively predict the next token in a large text corpus. All of our models are transformer decoders [VSP<sup>+</sup>17] in the style of GPT-3 [RNSS18, BMR<sup>+</sup>20].

We fine-tune pre-trained models using supervised learning to predict human labels on all of these tasks. Joint training means that there is no capability asymmetry between the base and critique models—thus we expect that any mistakes the base model “knows about” would also be “known” by the critique model.

We combine critiqueability tasks with answer “Yes” and critique tasks into a single training example (see Appendix A.5). Otherwise we have each example corresponding to a task, and shuffle all the examples for training. Note that our examples are not i.i.d. for multiple reasons: we have multiple questions per passage, the refinement demonstrations are collected at the same time as critique demonstrations, etc. See Appendix A for details.

Our models are trained for one epoch and we tune only the learning rate, with remaining hyperparameters fixed to be similar to pre-training.

We mask out all tokens except those corresponding to the human demonstrations. For example, in the critique task, we mask out the passage, topic, and answer being critiqued. See Appendix A.5 for details on input format.

#### Critiqueability and helpfulness score

Recall that for discrimination tasks, we collect binary yes/no labels. Rather than sampling binary labels from our models, we can look directly at logits to recover a probability. Thus we often use the terms critiqueability score and helpfulness score to refer to the quantity  $\frac{\text{Pr}[\text{Yes}]}{\text{Pr}[\text{Yes}] + \text{Pr}[\text{No}]}$  on the corresponding input.

On the critique task we “force” the model to give a critique even if the answer is perfect. Separately, the critiqueability score can be used to determine whether to ask it to critique in the first place, and the helpfulness score can be used to determine whether the critique is good after the fact.

#### Model scale

We use five pre-trained models with varying capabilities. Our pre-trained models are unfortunately not directly comparable to one another (for example, due to different pre-training datasets). However, on models which are directly comparable, the number of parameters correlates strongly with supervised fine-tuning validation loss. Using loss as the natural way to compare models of different architecture is suggested by [CCG<sup>+</sup>22], though here we use loss measured on fine-tuning instead of pre-training since it is the dataset commonality. Thus throughout the paper, we use “model scale” to refer to loss, measured in nats per token, and use that instead of model size for scaling laws [KMH<sup>+</sup>20].

## 2.3 Synthetic tasks

We also report results on four “synthetic” tasks, described in Table 3. For these tasks, we don’t require human data collection because we have binary ground truth for both answer and critique validity. We use hand-coded oracles for each of the base, critiqueability, critique, and helpfulness tasks.

Our tasks are chosen based on two criteria:

1. Evaluating critiques is easier than evaluating the base tasks.
2. The task is difficult but possible for most models. We tweak free parameters (e.g. sentence length for the unscramble task or number of digits for addition) to achieve this.

For our synthetic task models, we trained two rounds of models:

1. First we train on 100,000 generated base tasks with oracle demonstrations.
2. We then add 100,000 critiqueability task demonstrations, sub-sampled such that exactly half have incorrect answers, and 50,000 critique task demonstrations on that half. Answers are sampled from the first model at temperature 0, which we find improves accuracy. (We

|  | Base task description  | Critique task description   |
|--|--|---|
| <b>Addition</b>  | Add two 6-digit numbers  | A digit in the answer whose value is wrong, as well as the correct value for that digit (digits are indexed from least significant to most significant) |
| <i>Question:</i> 505579 + 900050<br><i>Answer:</i> 1505629<br><i>Critique:</i> Digit at index 6 should be 4  |  |   |
| <b>3-SAT</b>   | Given a satisfiable boolean formula in CNF form, output a satisfying assignment  | A clause that is not satisfied  |
| <i>Question:</i> Provide boolean values for $a, b, c, d, e, f, g, h, i$ that satisfy the following formula: $(\neg i \vee \neg f \vee e) \wedge (\neg e \vee \neg g \vee c) \wedge (g \vee \neg f \vee d) \wedge (\neg g \vee f \vee a) \wedge \dots$<br><i>Answer:</i> $a = \text{false}, b = \text{true}, c = \text{false}, d = \text{true}, e = \text{false}, f = \text{false}, g = \text{true}, h = \text{false}, i = \text{true}$<br><i>Critique:</i> The following clause is not satisfied: $(\neg g \vee f \vee a)$ |  |   |
| <b>Alphabetize</b>   | Given a list of 18 words, sort them in alphabetical order  | Either a missing/extra word in the resulting list, or a pair of adjacent words in the wrong order   |
| <i>Question:</i> Alphabetize the following words: growing prompts determining recreation evolve payable ruled patrols estimate emergency fate shrimp urges intoxicated narrator revert players pharmaceutical<br><i>Answer:</i> determining emergency evolve estimate fate growing intoxicated narrator patrols pharmaceutical payable players prompts recreation revert ruled shrimp urges<br><i>Critique:</i> Words misordered: evolve comes alphabetically after estimate   |  |   |
| <b>RACE</b>  | Provide the answers to two multiple choice questions about the same text passage. Questions are drawn from the RACE dataset [LXL <sup>+</sup> 17]. | Specify a question with a wrong answer, and give the correct answer   |
| <i>Question:</i> [passage]<br>Q1. Which one is the best title of this passage? A. Developing your talents. B. To face the fears about the future. C. Suggestions of being your own life coach. D. How to communicate with others.<br>Q2. How many tips does the writer give us? A. Two. B. Four. C. One. D. Three.<br><i>Answer:</i> 1 = C, 2 = D<br><i>Critique:</i> Answer to question 2 should be A   |  |   |

Table 3: Synthetic tasks with examples

occasionally repeat tasks when accuracy is so low or high that sub-sampling cannot guarantee uniqueness.)

This setup differs from the setup of topic-based summarization in two ways: (1) Each different model size is fine-tuned on a qualitatively different dataset in the second round. For topic-based summarization, different models are all trained on the same dataset. (2) We don’t do a third round of training on helpfulness tasks, although we do use the helpfulness oracle for evaluations.

### 3 Assisting critique finding

We ran experiments where our models assist human labelers at writing a set of critiques for answers. The assistance itself is a set of critiques shown to the labeler.

#### 3.1 Motivation

We chose this task because:

- Finding critiques is an important subtask of evaluating answer quality in general.
- We thought it would be the easiest task to use to measure the effect of model assistance. We initially tried a comparison-based task but it was more difficult to work with (see Appendix E).
- Suggesting critiques is a particularly natural form of assistance for critique-finding.