KIWI: A Dataset of Knowledge-Intensive Writing Instructions for Answering Research Questions

Fangyuan Xu[◇*] Kyle Lo[♡] Luca Soldaini[♡] Bailey Kuehl[♡] Eunsol Choi[◇] David Wadden[♡]
[◇]The University of Texas at Austin, [♡]Allen Institute for AI
{fangyuan, eunsol}@utexas.edu
{kylel, lucas, baileyk, davidw}@allenai.org

arXiv:2403.03866v1 [cs.CL] 6 Mar 2024

Abstract

Large language models (LLMs) adapted to follow user instructions are now widely deployed as conversational agents. In this work, we examine one increasingly common instruction-following task: providing writing assistance to compose a long-form answer. To evaluate the capabilities of current LLMs on this task, we construct **KIWI**, a dataset of knowledge-intensive writing instructions in the scientific domain. Given a research question, an initial model-generated answer and a set of relevant papers, an expert annotator iteratively issues instructions for the model to revise and improve its answer. We collect 1,260 interaction turns from 234 interaction sessions with three state-of-the-art LLMs. Each turn includes a user instruction, a model response, and a human evaluation of the model response. Through a detailed analysis of the collected responses, we find that all models struggle to incorporate new information into an existing answer, and to perform precise and unambiguous edits. Further, we find that models struggle to judge whether their outputs successfully followed user instructions, with accuracy at least 10 points short of human agreement. Our findings indicate that **KIWI** will be a valuable resource to measure progress and improve LLMs' instruction-following capabilities for knowledge intensive writing tasks.[1]

1 Introduction

As LLM-powered conversational agents (OpenAI, 2023; Touvron et al., 2023; Google, 2023) have gained widespread adoption, users have turned to these systems for assistance on a wide range of real-world tasks. In particular, recent works studying LLM interactions "in the wild" (Ouyang et al., 2023; Zheng et al., 2023a) have discovered that 10-20% of user queries contain requests for writing assistance, i.e. using an LLM to create, revise, or organize a piece of written text. However, we currently lack an understanding of LLMs' capabilities as writing assistants, especially in knowledge-intensive settings. What types of writing instructions do users issue to LLMs, and how well can LLMs follow different types of instructions?

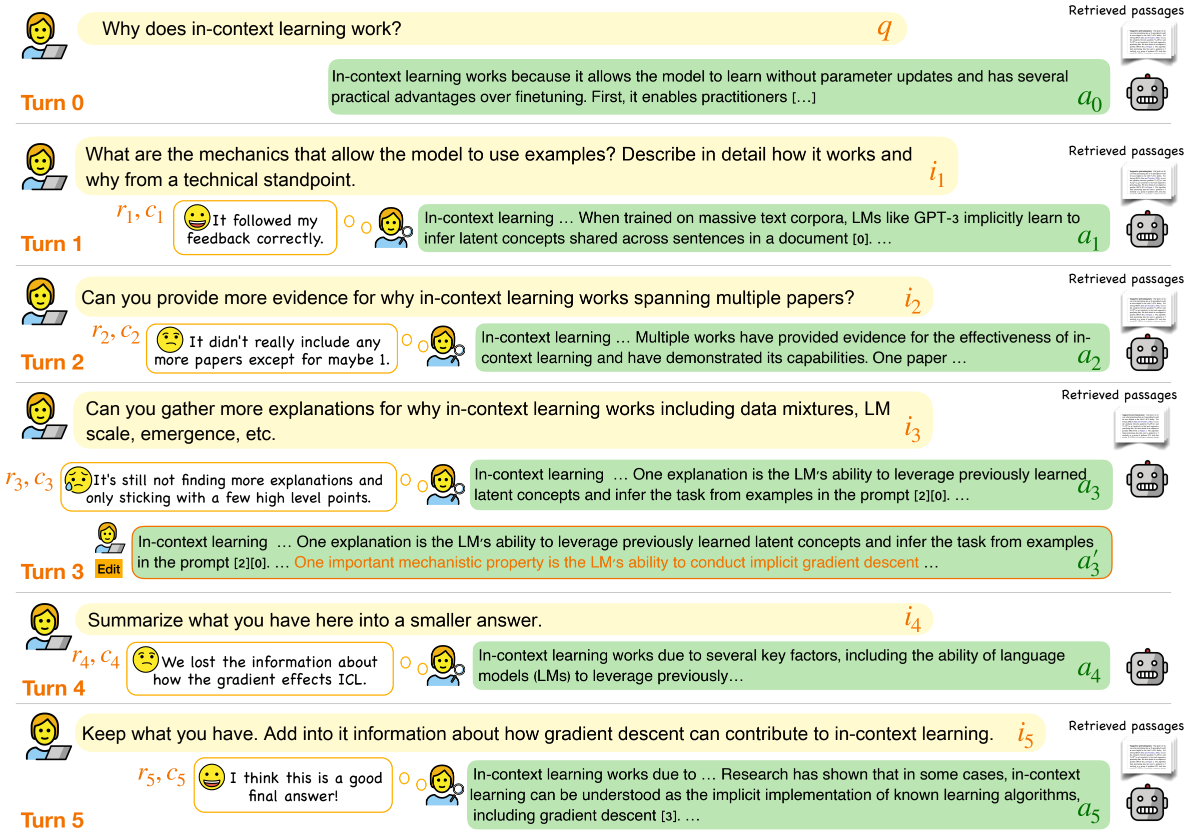
We present **KIWI**, a dataset of expert-written **K**nowledge-**I**ntensive **W**riting **I**nstructions to better understand LLMs' instruction-following abilities as writing assistants. To collect **KIWI**, we set up an interactive interface between a researcher and an LLM (Figure 1). We first prompt an LLM to generate a long-form answer (Fan et al., 2019) to an NLP research question, based on a set of passages from relevant research papers. A researcher then iteratively issues instructions for the model to revise the answer and evaluates the model-generated revision. The interaction session continues until the user is satisfied with the final answer or a maximum number of turns is reached. **KIWI** contains instructions collected from 234 interaction sessions with three state-of-the-art language models (GPT-4,GPT-3.5-turbo and LLaMA-2-70b-chat), providing a total of 1,260 unique instruction instances. Each instance consists of (1) a user instruction, (2) the model's previous and (3) revised answers, and (4) a human annotator's judgment (both categorical rating and free-form explanation) on whether the revised answer followed the instruction.

Using **KIWI**, we conduct an in-depth analysis to characterize the types of instructions issued by researchers, and to measure how well models can follow different types of instructions. We find that LLMs *do not* excel at this task yet, with the best model (GPT-4) achieving success for only 59% of the instructions. Specifically, LLMs fail to precisely follow user's instructions (such as satisfying location and length constraints), often cause answer quality to degrade when integrating new information into a previous answer, and struggle to avoid making changes to answers that are not requested.

---

[*] Work performed during an internship at AI2.

[1] Our dataset is released at `https://www.cs.utexas.edu/~fxu/kiwi/`.



Why does in-context learning work?

In-context learning works because it allows the model to learn without parameter updates and has several practical advantages over finetuning. First, it enables practitioners [...]

What are the mechanics that allow the model to use examples? Describe in detail how it works and why from a technical standpoint.

It followed my feedback correctly.

In-context learning … When trained on massive text corpora, LMs like GPT-3 implicitly learn to infer latent concepts shared across sentences in a document [0]. …

Can you provide more evidence for why in-context learning works spanning multiple papers?

It didn't really include any more papers except for maybe 1.

In-context learning … Multiple works have provided evidence for the effectiveness of in-context learning and have demonstrated its capabilities. One paper …

Can you gather more explanations for why in-context learning works including data mixtures, LM scale, emergence, etc.

It's still not finding more explanations and only sticking with a few high level points.

In-context learning … One explanation is the LM's ability to leverage previously learned latent concepts and infer the task from examples in the prompt [2][0]. …

In-context learning … One explanation is the LM's ability to leverage previously learned latent concepts and infer the task from examples in the prompt [2][0]. … One important mechanistic property is the LM's ability to conduct implicit gradient descent …

Summarize what you have here into a smaller answer.

We lost the information about how the gradient effects ICL.

In-context learning works due to several key factors, including the ability of language models (LMs) to leverage previously…

Keep what you have. Add into it information about how gradient descent can contribute to in-context learning.

I think this is a good final answer!

In-context learning works due to … Research has shown that in some cases, in-context learning can be understood as the implicit implementation of known learning algorithms, including gradient descent [3]. …

Figure 1: An example interaction session between a user and our system. Given a question q and a set of documents $\mathbf{D}$, the model first generates an initial answer a_0. At each turn t, the user issues an instruction i_t, for which an updated answer a_t is generated. The user provides a rating r_t and a comment c_t for the generated answer, and optionally edits the answer (such as a_3' in turn 3) such that the edited answer completely follows the instruction.

Finally, we examine the ability of the strongest LLM (GPT-4) to evaluate whether a response follows an instruction, comparing its judgement to the user ratings in **KIWI**. Unlike prior works which found that LLMs can judge outputs as reliably as humans for open-ended instruction following (Zheng et al., 2023b; Li et al., 2023), our experiments show that GPT-4 cannot reliably evaluate responses for instructions in **KIWI**, which are often specific and precise. While the best performing model which retrieves and prepends 10-shot examples improves upon zero-shot performance by 5% accuracy, it still lags behind human agreement by 12% accuracy. Given these findings, we believe that **KIWI** will be a valuable resource for future research in instruction following and evaluation.

2 Interaction Design

2.1 Design goals

Unlike prior works (Kopf et al., 2023; Zhao et al., 2023; Zheng et al., 2023a) which deploy an LLM in the wild to collect diverse types of user interaction data, we focus on the use case of seeking writing assistance. We design a knowledge-intensive interaction setting with two key considerations. First, the information in the answer should be grounded to a set of relevant documents. This allows us to examine model's ability to leverage information from *multiple* documents, motivated by real-world writing tasks such as literature review (Shen et al., 2023). Second, the user interacts with the model for *multiple* turns to iteratively revise the answer. Instead of collecting a single editing instruction for (question, initial answer) pairs, this setting not only adheres to the iterative nature of the text editing process (Collins and Gentner, 1980; Du et al., 2022), but also allows us to collect diverse and fine-grained instructions across different editing stages as we will see in §4.1.

2.2 Interaction process

We provide an overview of the interaction process (Figure 1) and describe the data collection procedure in §3. The system inputs are an NLP research question q and a collection $\mathbf{D}$ of research papers which, collectively, contain information sufficient to write a high-quality, paragraph-length answer to

q. To begin the interaction process, an LLM generates a "first draft" answer a_0 to q based on $\mathbf{D}$. Each interaction turn t involves four steps with actions taken by the user (👤) or the model (💻): (1) The user reads the LLM-generated answer and issues an instruction i_t to the model to revise the answer. (2) The model then generates an updated answer a_t given q, a_{t-1} and i_t. (3) The user provides a rating r_t and a comment c_t evaluating how well a_t follows i_t. (4) The user can choose to edit the answer a_t if it did not follow their instruction, producing a'_t. Afterwards, the user can start a new turn by issuing another instruction for a_t (or a'_t). The interaction continues until either the user is satisfied with the answer, or a total of 10 turns is reached. We describe each step below with additional details (prompts, hyperparameters) in Appendix A.1.

💻 **Generating the initial answer** Given the question q, we first retrieve a set of five passages from $\mathbf{D}$ with a retrieval model R, denoted as $R(q, \mathbf{D})$. We prepend the retrieved passages to the question q to generate the initial answer a_0 with a language model M, that is, $a_0 = M(q, R(q, \mathbf{D}))$.[2]

👤 **Issuing instructions** To start a turn t, the user issues an instruction i_t, specifying how they would like the LLM to revise its answer in the previous turn a_{t-1}, and categorizes its type I_t. We identify two types of instructions. **Information-seeking** instructions require the model to incorporate new information into the answer; e.g. Turn 3 of Figure 1 asks the model to acquire additional information from the source documents $\mathbf{D}$. **Stylistic** instructions ask the model to transform content that is already present; e.g. Turn 4 in Figure 1 asks the model to compress its answer. For each interaction turn, the user indicates the type of the instruction I_t by selecting a checkbox in the annotation interface.

💻 **Answer revisions** Next, the model M is prompted to revise its previous answer a_{t-1} to satisfy user instruction i_t. We use one prompt for information-seeking instructions (which require passage retrieval from $\mathbf{D}$), and another for stylistic questions (which do not).

For information-seeking instructions, we first leverage a retrieval model R to retrieves five passages p_t from $\mathbf{D}$, using the instruction i_t as the query. Next, we prompt M to summarize the information in p_t that is likely to be relevant given (a_{t-1}, i_t); we denote this summary as s_t. Finally, we generate an updated answer by prompting M with $(q,\ a_{t-1},\ i_t,\ s_t)$.[3] For stylistic instructions, we generate the updated answer directly by providing $(q,\ a_{t-1},\ i_t)$ as input. We only include a single previous answer in the prompt to generate the updated answer; in the pilot study, we found that this approach yielded comparable performance to including the entire interaction history.

👤 **Measuring answer quality** After a revised answer a_t is generated, the user provides a rating indicating whether M successfully followed their instruction: $r_t = \mathbf{r}(i_t, q, a_{t-1}, a_t)$. We instruct the annotators to rate a_t as one of the following:

- `good`: Successfully followed the instruction.
- `neutral`: Partially followed the instruction.
- `bad`: Didn't follow the instruction at all, or modified the answer to an undesirable state.

Since our goal is to measure the helpfulness of the model response as determined by the annotator, we ask annotators to use their own judgement in selecting a rating rather than providing a detailed guideline.[4] Nonetheless, we find in §5 that annotators have high agreement on this categorization. We also collect a mandatory free-form comment c_t from the annotator explaining their rating r_t. Example turns are in Table 8 in the Appendix A.2.

👤 **Handling incorrect instruction-following** When $r_t \neq$ `good`, answer a_t does not fully follow instruction i_t and needs to be corrected. While the user might issue instructions in the subsequent turns to correct what has gone wrong with a_t,

the success of the user's correction attempt depends heavily on the model's instruction following ability. The user might fall into a loop of continuing to prompt a model which is not able to perform the action requested in the instruction. To avoid such scenarios, we allow, but do not require, the user to edit a_t when $r_t \neq$ `good`, which induces a'_t, such that $\mathbf{r}(i_t, q, a_{t-1}, a'_t) =$ `good`. If the user chooses to not edit the answer and $r_t =$ `bad`, we remove the answer from the answer history when generating answers in the subsequent turns.

---

[2] We perform retrieval because prepending the full text of multiple papers (each with around 6,000 tokens) exceeds the context window of the LLMs (with at most 8,000 tokens).

[3] In preliminary experiments, we found that this "summarize-then-answer" workflow yielded better results than directly prompting M with the retrieved passages p_t.

[4] To alleviate annotator burden, annotators are not required, but welcomed, to check the factual correctness of model-generated answers. Assessing faithfulness of LLM generations represents an important but orthogonal research direction.

3 Dataset Collection

We first describe our approach to curate high-quality $(q, \mathbf{D})$ pairs (§3.1), followed by our annotation protocol for interaction collection (§3.2).

3.1 (Question, Document set) creation

The process proposed in §2.2 requires high-quality $(q, \mathbf{D})$ pairs to serve as a starting point for interaction collection. We aim to create $(q, \mathbf{D})$ pairs which satisfy the following desiderata: (1) the questions should be realistic and likely to be asked by a researcher, (2) the questions should be challenging, requiring information from multiple documents and (3) each document set $\mathbf{D}$ should contain sufficient information to answer its corresponding question. We observe that the related work section of a research paper often answers a set of implicit research questions, and supplies the documents answering each question as citations. Thus, we collect high-quality $(q, \mathbf{D})$ pairs by writing questions implied by related work sections, and pairing each question with its cited papers.

Source article selection We select a set of roughly 100 papers published in ACL 2023 across 11 different tracks listed in the ACL handbook.[5] This set of papers covers a wide range of NLP research topics and was published after the training data cutoff time of the language models used for data collection in this work. The model thus has to answer the question based on the documents, instead of memorizing from its training data.

Question creation and filtering Three of the authors with prior NLP research experience manually annotate 88 questions from 75 papers. Given a source paper, the annotator first reads through its related work section and decides if a good question can be derived. A good question should fulfill the following criteria: (1) it should "stand alone", i.e. it is not anchored in the source paper; and (2) there should be at least four articles cited in the related work which contain relevant information for answering the question. If a good question can be derived, the annotator writes a question q, together with the list of evidence papers $\mathbf{D}$ and the section of the related work r, from which the question is derived. We query the S2ORC (Lo et al., 2020) corpus to retrieve the full text of each paper and extract paragraphs. To ensure sufficient coverage, we filter out questions with either (1) more than one evidence paper missing in S2ORC or (2) fewer than 4 evidence papers in total. We obtain a total of 78 questions with an average of 6 cited papers per question.

Source Paper: Small pre-trained language models can be fine-tuned as large models via over-parameterization (Gao et al., 2023)
Related work paragraph: Over-parameterization in Neural Network. Over-parameterization has shown the superiority on providing better model initialization (Arpit and Bengio, 2019), improving model convergence (Du et al., 2019; Allen-Zhu et al., 2019b; Gao et al., 2022a) and generalization (Allen-Zhu et al., 2019a). [...] **Question**: Why does over-parameterization lead to improved initialization and convergence of deep neural networks? **Evidence papers**: Gao et al. (2021); Allen-Zhu et al. (2018); Li and Liu (2016); Du et al. (2018)

Table 1: An example $(q, \mathbf{D})$ annotation (§3.1). Given a source paper, the annotator first selects the related work paragraph(s) that imply the question, then writes the question and extracts evidence papers cited.

Analysis of collected questions Our annotated questions consist of 12 words on average, with the majority of questions starting with "How" (47%) and "What" (46%). Upon manual examination, all models are able to generate a reasonable initial answer given the question and retrieved passages. This shows that our pipeline creates high quality data: the document set indeed contains sufficient information to answer the question.[6] See example questions in Table 7 in the Appendix A.2.

3.2 Interaction data collection

At the start of an interaction session, the annotator is shown the question, initial answer, and the titles of relevant papers for the question. During the interaction, the annotator is shown the full interaction history, and the retrieved passages from the last info-seeking instruction. We release annotation instruction[7] and provide annotation interface screenshot as Figure 4 in Appendix A.1.

Language models We conduct annotations with three models M: GPT-3.5-turbo, GPT-4 and

---

[5] We choose to focus on NLP research questions, since the authors of this paper have expertise in this domain.

[6] In the initial phase of the project, we explored using GPT-4 to generate questions given related work paragraphs, but found that the generated questions were often not answerable by the cited papers. We include analysis on question generation in the Appendix A.6.

[7] `https://docs.google.com/document/u/1/d/e/2PACX-1vTrsM5r-p5kjy6Ue7AkbLqwOmJ-GoK9kwToBEnW-Nwk00yd4tKSFkWu9p63j7rH-PvNwwNXLzTyRWhi/pub`

LLaMA-2-chat(70B) (Touvron et al., 2023), competitive commercial and public LLMs available at the time of data collection, respectively. For all models, we decode with temperature 0.7. During data collection, the identity of model M is not revealed to the annotator. For each question, we collect one interaction session with each model. We ensure that annotators do not answer the same question more than once.

Retrieval models We use the state-of-the-art dense retriever Contriever (Izacard et al., 2021) finetuned on MSMARCO (Campos et al., 2016) as our retriever R. [8]

Annotators To ensure our annotators have expertise in answering research questions, we recruit NLP researchers through Upwork[9] and professional networks. We pay the annotators USD $25 per hour on average. The annotators first participate in a paid pilot study, which involves reading the instructions and completing an annotation example. A total of 22 annotators participated in our pilot annotations and 15 of them performed final annotations for **`KIWI`**. The final group of annotators consists of four people with a Ph.D. degree, seven NLP Ph.D. students, one person with a Master's degree and three undergraduate students. All annotators have at least one year of NLP research experience. On average, each interaction session required 15 minutes to complete. To ensure that the annotators are familiar with the subject matter of the question, we first collect their preferences on a set of topics derived from the track information of each question's source paper. We assign questions to annotators based on their preferences. The annotations were collected from July to December 2023. In total, our annotations cost $2,240 USD.

4 Analysis of `KIWI`

We first present results on overall model performance (§4.1). Then we conduct a fine-grained analysis to understand the different types of instructions (§4.2) and errors made by LLMs (§4.3).

4.1 Dataset overview

We present statistics of the interaction sessions in Table 2. We observe that the number of turns required to reach a satisfactory answer negatively correlates with model capability, with `gpt-4` requiring the fewest turns and `llama2` requiring the most. Comparing instruction types, around 60% of instructions are information-seeking for `gpt-4` , and 45% for `llama2` ; this suggests that the `gpt-4` is able to more quickly generate stylistically acceptable text, allowing users to focus on information content. We also observe that users tend to issue information-seeking instructions early in the interaction session followed by stylistic instructions (Figure 2). This confirms that collecting instructions over multiple turns allows us to cover diverse instructions occurring in different writing stages.

Model	**# turn (edited)**	**avg. # t/s**	**% info/style**
`gpt-4`	370 (55)	4.7	59/41
`gpt-3.5`	402 (111)	5.2	61/39
`llama2`	488 (149)	6.3	44/56
Total	1,260 (315)	5.4	54/46

Table 2: Data statistics of the **`KIWI`**. We collect a total of 234 sessions (78 sessions per model). We report the total number of turns, number of turns with edited answers, number of turns per session, and the distribution of instruction types for each model.

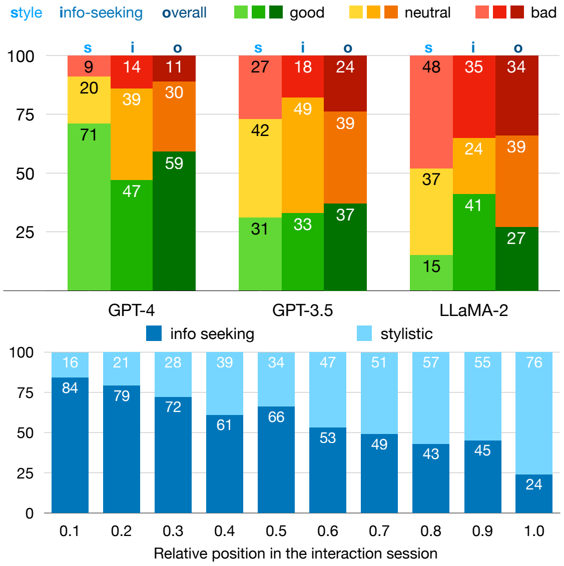


Figure 2: Top: distribution of annotator ratings. The left columns represent ratings for stylistic instructions (s), the middle columns for info-seeking instructions (i) and the right columns for overall (o) ratings. Bottom: distribution of user-issued instructions across the session.

`KIWI` presents a challenge for existing LLMs: The distribution over user ratings of model responses is shown in Figure 2. `gpt-4` exhibits the strongest instruction-following performance (59%

---

[8] In preliminary experiments, we found that Contriever outperformed sparse methods like BM25 (Robertson and Zaragoza, 2009), which struggle at handling synonyms.

[9] `https://www.upwork.com/`

Type	% info/style	% good	Definition	Example Instruction
More information (43%)	**88**/12	֍: **57** ֍: 45 ∞: 34	Asking for auxiliary information related to the question or the answer (e.g. example, results, limitation)	How well do state-of-the-art models perform on these datasets?
Expand coverage (9%)	**86**/14	֍: **40** ֍: 31 ∞: 12	Asking for more information directly answering the question, usually from multiple papers.	Keep the text content as it is, but extract examples of applications of contrastive explanations to NLP from papers 0,1, and 2.
Clarification (5%)	**69**/31	֍: **52** ֍: 50 ∞: 50	Asking for clarification of a concept already in the answer	Do all of the quantization methods you mention require additional training?
Remove (13%)	4/**96**	֍: **77** ֍: 28 ∞: 16	Requests to remove certain part of the answer	Remove the repeated techniques listed in the last paragraph.
Specific edit (6%)	23/**77**	֍: **59** ֍: 35 ∞: 19	Requests to directly add information that are provided in the instruction verbatim to the answer	At the end of the final paragraph, add the sentence "Evaluation of factuality for natural text remain active research lines."
Condense (10%)	4/**96**	֍: **63** ֍: 27 ∞: 23	Requests to shorten or summarize the answer	Please shorten the answer to a maximum of 6 paragraphs.
Reorganize (11%)	13/**87**	֍: **74** ֍: 27 ∞: 34	Requests to reorganize the answer (e.g. move things around)	Reorganize the answer so that positive results are described first, followed by challenges and open problems
Others (3%)	**53**/47	֍: **62** ֍: 23 ∞: 29	Edits that don't belong to any of the category above	Don't change anything further about this answer but copy it as-is.

Table 3: Definitions and example instructions for the fine-grained instruction types (§4.2). We report the distribution classified by GPT-4 and % good rating (color coded) across the three models (֍ `gpt-4`, ֍ `gpt-3.5`, ∞ `llama2`).

good responses), while `llama2` exhibits the weakest (27% good responses). Interestingly, trends for the two types of instruction differ—while `gpt-4` provides good responses to stylistic instructions around 70% of the time, it struggles at following information seeking instructions (i.e. adding new information to an answer). In contrast, `llama2` performs extremely poorly at following stylistic feedback, with only 15% of responses rated as good. All models fail to follow user instructions in more than 40% of responses, leaving significant headroom for LLMs' instruction-following ability.

4.2 Fine-grained instruction analysis

Based on our findings that LLMs struggle at following the instructions in **`KIWI`**, we conduct a fine-grained analysis to categorize the different types of instructions found in the dataset, and measure model performance on each. We manually label a sampled set of instructions, finding eight categories, and then scale up using GPT-4. We construct a few-shot prompt containing [(original question, instruction) → fine-grained instruction type] demonstrations for each instruction type, and prompt GPT-4 to label the entire dataset. Details about the prompt and human analysis can be found in the Appendix A.3. We validate our approach by confirming that GPT-4 achieves 92.5% agreement with humans on a held-out set of 40 instructions.

`KIWI` contains diverse instruction types: The results of our analysis are shown in Table 3. Although we did not reference coarse-grained category labels when defining the fine-grained categories, we observe that our resulting fine-grained categories can be divided based on our distinction between information-seeking and stylistic instructions. Information-seeking instructions include requests to provide auxiliary information, expand answer coverage, or provide clarification. Stylistic instructions range from lower-level requests such as performing a specific edit, to more complicated operations such as condensing and reorganizing the answer. We report additional statistics for each category in Table 6 in the Appendix A.3.

Models struggle at integrating new information and precise editing: Among the information seeking instructions, "expand coverage" is the most difficult, as this type of instruction requires both successful retrieval of relevant passages from *multiple* papers and coherent integration into the previous answer. For stylistic instructions, `gpt-4` performs well at removal and reorganization, achieving more than 70% good ratings, while `gpt-3.5` and `llama2` perform poorly. Interestingly, all three models, including `gpt-4` , struggle to make specific edits (e.g. add a requested sentence verbatim), which are usually trivial for humans. While prior work has found that LLMs excel at single document text summarization (Goyal et al., 2022), we observe poor performance for the "condense" instructions (which request to shorten previous answer), with `gpt-4` 's response rated as good only 63% of the time. This is perhaps because answers in our task involve information from *multiple* papers, and multi-document summarization remains a challenge for current LLMs (Shaib et al., 2023).

4.3 Error analysis

To better understand *how* models fail to follow instructions, we analyze the free-form comments c_t written by the annotators. As in §4.2, we manually label a small set of comments and scale up with GPT-4 to all 760 turns which received a `neutral` or `bad` rating. We construct a prompt containing [(instruction, comment → target category)] demonstrations for each fine-grained error category; see Appendix A.4 for details of human analysis and the exact prompt. On a validation set with 22 examples, GPT-4 achieves 90% agreement with humans.

We find five major categories (Table 4). The most common error type is **Unrequested change**, indicating that models fail to maintain answer consistency across turns. Models also fail to satisfy hard constraints, such as the location ("At the beginning of the answer...") and length of the information ("Add one sentence about...") specified in the instruction. This suggests that current models struggle to perform *precise* actions, in agreement with findings from recent work (Sun et al., 2023) on controlled generation. Finally, models struggle to coherently integrate new information into the existing answer text, often leading to imbalanced structure or awkward answer flow.

Retrieval analysis The model could fail to follow the instruction due to retrieval failure instead of the language model. We randomly sample 20 turns with info-seeking instructions that are rated as `neutral` or `bad` and manually examine retrieval performance. We find that for 50% of the turns, relevant passages are retrieved which contain sufficient information to follow the instruction, yet the language models fail to integrate the information into the previous answer. For 20% of turns, some (but not all) relevant passages are retrieved. For the remaining 30%, the retrieval system fails to retrieve any relevant information. This demonstrates room for improvement in *both* components.

Error type	**Example**
Unrequested change (31%) The update introduces changes not requested.	**Instruction**: Limit the number of sentences describing advantages and disadvantages for each method to 3. **Comment**: The model shortened the desired paragraphs but added another and removed information from the previous answer.
Ignored (21%) The requested change was not made.	**Instruction**: Reduce the discussion about task-specific pretraining objectives **Comment**: Model does not seem to have reduced any information at all.
Constraint failure (11%) The update in the answer does not follow some constraint(s).	**Instruction**: As first sentence of the text provide a definition of "clarification question". **Comment**: It gave the definition but not as first sentence in the text.
Poor integration (9%) The answer is less coherent after the update.	**Instruction**: Please start the answer by describing what "leveraging future utterance for dialogue generation" is. **Comment**: The requested info was added. However, the quality of the answer degraded somewhat and now has a odd ordering.
Others (28%) The update is not satisfactory for other reason.	**Instruction**: Explain some examples of methods in detail. **Comment**: only one example of an actual technique is given here

Table 4: Fine-grained error categories alongside example instruction and corresponding annotator comment pairs. We report the distribution classified by GPT-4 on interaction turns that are rated as `neutral` or `bad`.

5 Experiments: Automatically evaluating instruction following

Our analysis so far demonstrates that current LLMs often fail to follow users' instructions to revise long-form answers. Next, we examine whether they can serve as an *evaluator* to assess whether a model-generated answer followed the instructions (Dubois et al., 2023; Chiang and Lee, 2023; Zeng et al., 2023; Zheng et al., 2023b; Liu et al., 2023).

Setting and data Formally, the task is to predict the user rating $r_t = \mathbf{r}(q, i_t, a_{t-1}, a_t)$, introduced in §2.2. To simplify the task, we collapse {`neutral`, `bad`} into a single label. We randomly split the 1,260 turns into train/dev/test set, resulting in 888 training, 186 validation and 186 test instances.

Metrics and baselines We evaluate using **Acc**uracy, **P**recision, **R**ecall and **F1** against the collected human labels r_t, with the good label as the positive class. We report two baselines: a **Random** baseline which randomly assigns a label accord-

Model	Acc	P	R	F1	% g/b
Majority	0.53	0.0	0.0.	0.0	0/100
Random	0.47	0.42	0.31	0.36	38/62
zero-shot	0.63	0.57	**0.91**	**0.70**	75/25
one-shot	0.62	0.50	0.86	0.65	65/35
ten-shot	**0.68**	**0.67**	0.59	0.63	40/60
T5 (finetuned)	0.64	0.5	0.73	0.37	24/76
Human*	0.80	0.63	0.90	0.75	46/54

Table 5: Test set results on automatic evaluation for instruction following edits. We also report the % of **g**ood and **b**ad ratings in the model predictions.

ing to the training data distribution, and a **Majority** baseline which always chooses the majority class in the training data. We measure **Human** agreement by collecting a second set of human ratings for 65 randomly-sampled instructions, and measuring agreement with the original ratings.[10]

Model We experiment with **zero-shot** and **few-shot** prompting with `gpt-4` and a **fine-tuned** T5-large (770M) (Raffel et al., 2019) model. For zero-shot prompting, we construct an instruction which specifies the criteria for the two ratings. Inspired by prior work on retrieving in-context examples (Rubin et al., 2022), we retrieve the k turns from the training data whose instructions have the highest BM25 similarity with the test instruction. See Appendix A.5 for details.

Results We report results on the test set in Table 5. GPT-4 zero-shot performs slightly worse than choosing the majority label; we found that it is biased heavily towards judging answers as good. Adding 10 in-context examples improves performance, while leaving a substantial gap relative to human agreement. Finetuned T5 only achieves similar accuracy with zero-shot `gpt-4` . While recent work found that LLMs can rate responses as reliably as human (Chiang and Lee, 2023; Zheng et al., 2023b), our experiments show that they are not reliable for judging responses for specific instructions (Zeng et al., 2023; Liu et al., 2023).

6 Related Work

Instruction following A number of recent efforts have collected instruction following datasets to train and evaluate LLMs. Some datasets contain crowd-sourced interactions in the wild (Taori et al., 2023; Conover et al., 2023; Kopf et al., 2023; Zhao et al., 2023; Zheng et al., 2023a), while others target specific tasks such as summarization (Liu et al., 2023) and controllable generation (Zhou et al., 2023). Our work focuses specifically on instruction-following for writing assistance and text editing. A number of prior works have curated instruction-based editing benchmarks (Dwivedi-Yu et al., 2022; Shu et al., 2023; Raheja et al., 2023; Zhang et al., 2023). However, these efforts consist of post-hoc synthetic instructions derived from previous datasets, consisting of a limited set of edit instructions. Our data consists of diverse instructions issued by expert users interacting with LLMs.

LLM-based Evaluation Recent works have investigated using an LLM as an evaluator in place of human evaluation (Chiang and Lee, 2023; Zheng et al., 2023b; Li et al., 2023) for instruction following. A line of work shows that LLMs can judge responses as reliable as human annotators (Zheng et al., 2023b; Li et al., 2023). Recent work (Zeng et al., 2023) points out that evaluating open-ended instruction following exhibit large subjectivity and thus low human performance. The authors curate a set of instructions that are objective and craft adversarial response to stress test LLMs, finding that the models struggle at evaluating responses for such instructions while humans exhibit high agreement. Our study also shows that LLMs struggle at evaluating responses for precise instructions collected in a *realistic* setting, complementing prior work.

7 Conclusion

We present **`KIWI`**, a dataset with expert-issued writing instructions to revise a long-form answer to a research question using relevant documents. Our analysis and experiments show that current LLMs (including GPT-4) cannot perform this task reliably yet and identify common failure patterns. We are optimistic that **`KIWI`** will be a useful resource for several research directions. First, the model revisions that were judged as successful by humans, and answers edited manually by humans when models failed, can be used as training data to improve models' instruction following abilities. Second, human judgments of model revision quality can be used to develop more accurate reward models for writing assistance. Finally, the instructions in **`KIWI`** can be used as inputs to evaluate the performance of future models; either through human judgments, or by future reward models.

---

[10] Two authors and two annotators from the interaction annotation performed the annotation. We ensure the interaction annotators do not re-annotate their own previous interactions.

Acknowledgments

We thank Nicholas Behrje, Sergey Feldman, Sebastian Joseph, Yoonjoo Lee, Clara Na, Matthias Rémy, Prasann Singhal, Zayne Sprague, Irina Temnikova, Albert Yu and other annotators for participation in creating the dataset. We thank the Semantic Scholar team and UT Austin NLP lab for helpful discussion and piloting the annotations. We thank Pradeep Dasigi, Sachin Kumar, and Leo Zeyu Liu for feedback to improve the paper draft.

Author Contributions: Fangyuan Xu led the project and performed all the technical contributions including designing and building the annotation interface, dataset collection, running the experiments and performing the data analysis. Fangyuan also contributed to project scoping and ideation and led the writing of the paper. Bailey Kuehl helped with collecting the instruction data, including piloting the task, giving feedback and hiring and managing the Upwork annotators. David Wadden, Kyle Lo and Luca Soldaini were mentors of the project during and after Fangyuan's internship, contributed to project scoping, experiments design, ideation and direction throughout the project. David Wadden and Kyle Lo contributed to paper writing and Luca Soldaini contributed to annotation interface development. Eunsol Choi provided mentorship throughout the project, including project scoping, ideation, experiement design and paper writing.

Ethics Statement

Our annotation protocol has been determined to be exempted from review by an IRB board at UT Austin. All data collected will be made publicly available under the MIT license. The data collection process did not require any information that can be used to uniquely identify individual annotators. We manually examine the annotation and make sure no such information or offensive content is present in the data collected.

Limitations

Our study focus on the domain of writing answers for scientific questions and more specifically, NLP research questions. While we believe some of our findings could generalize to other domains such as creative writing, our data might only represent a subset of instructions that occur in other writing tasks or domain. We only cover the English language, and future work might collect similar data for other languages or look into leveraging our data for cross-lingual transfer.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2018. A convergence theory for deep learning via over-parameterization. *ArXiv*, abs/1811.03962.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268.

Cheng-Han Chiang and Hung-Yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Annual Meeting of the Association for Computational Linguistics*.

Allan Collins and Dedre Gentner. 1980. *A framework for a cognitive theory of writing*, pages 51–72. Erlbaum.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Simon Shaolei Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. 2018. Gradient descent provably optimizes over-parameterized neural networks. *ArXiv*, abs/1810.02054.

Wanyu Du, Vipul Raheja, Dhruv Kumar, Zae Myung Kim, Melissa Lopez, and Dongyeop Kang. 2022. Understanding iterative revision from human-written text. *ArXiv*, abs/2203.03802.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *ArXiv*, abs/2305.14387.

Jane Dwivedi-Yu, Timo Schick, Zhengbao Jiang, Maria Lomeli, Patrick Lewis, Gautier Izacard, Edouard Grave, Sebastian Riedel, and Fabio Petroni. 2022. Editeval: An instruction-based benchmark for text improvements. *ArXiv*, abs/2209.13331.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Tianxiang Gao, Hailiang Liu, Jia Liu, Hridesh Rajan, and Hongyang Gao. 2021. A global convergence theory for deep relu implicit networks via over-parameterization. *ArXiv*, abs/2110.05645.

Ze-Feng Gao, Kun Zhou, Peiyu Liu, Wayne Xin Zhao, and Ji rong Wen. 2023. Small pre-trained language models can be fine-tuned as large models via overparameterization. In *Annual Meeting of the Association for Computational Linguistics*.

Google. 2023. Gemini: A family of highly capable multimodal models. *ArXiv*, abs/2312.11805.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Andreas Kopf, Yannic Kilcher, Dimitri von Rutte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Rich'ard Nagyfi, ES Shahul, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. *ArXiv*, abs/2304.07327.

Fengfu Li and Bin Liu. 2016. Ternary weight networks. *ArXiv*, abs/1605.04711.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yixin Liu, A. R. Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq R. Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. 2023. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization. *ArXiv*, abs/2311.09184.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

Siru Ouyang, Shuo Wang, Yang Liu, Ming Zhong, Yizhu Jiao, Dan Iter, Reid Pryzant, Chenguang Zhu, Heng Ji, and Jiawei Han. 2023. The shifted and the overlooked: A task-oriented investigation of user-gpt interactions. In *Conference on Empirical Methods in Natural Language Processing*.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. Coedit: Text editing by task-specific instruction tuning. *ArXiv*, abs/2305.09857.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Chantal Shaib, Millicent Li, Sebastian Joseph, Iain Marshall, Junyi Jessy Li, and Byron Wallace. 2023. Summarizing, simplifying, and synthesizing medical evidence using GPT-3 (with varying success). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1387–1407, Toronto, Canada. Association for Computational Linguistics.

Zejiang Shen, Tal August, Pao Siangliulue, Kyle Lo, Jonathan Bragg, Jeff Hammerbacher, Doug Downey, and Joseph Chee Chang. 2023. Beyond summarization: Designing ai support for real-world expository writing tasks. *ArXiv*, abs/2304.02623.

Lei Shu, Liangchen Luo, Jayakumar Hoskere, Yun Zhu, Canoee Liu, Simon Tong, Jindong Chen, and Lei Meng. 2023. Rewritelm: An instruction-tuned large language model for text rewriting. *ArXiv*, abs/2305.15685.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Frederick Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan

Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *ArXiv*, abs/2310.07641.

Haopeng Zhang, Hayate Iso, Sairam Gurajada, and Nikita Bhutani. 2023. Xatu: A fine-grained instruction-based benchmark for explainable text updates. *ArXiv*, abs/2309.11063.

Wenting Zhao, Xiang Ren, Jack Hessel, Yejin Choi Claire Cardie, and Yuntian Deng. 2023. (inthe)wildchat: 650k chatgpt interaction logs in the wild.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Haotong Zhang. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *ArXiv*, abs/2309.11998.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685.

Wangchunshu Zhou, Yuchen Jiang, Ethan Gotlieb Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. *ArXiv*, abs/2304.14293.

A Appendix

A.1 Implementation details for the interaction system

We implement the annotation interface with gradio[11]. Figure 4 presents a screenshot of our annotation interface. We use the OpenAI API[12] for gpt-family model inference and the Together AI API [13] for `llama2`. Below are the prompt templates we use.

Prompt for generating initial answer We include the top 5 retrieved passages from the document set to generate the initial answer. `title_i` and `passage_i` refers to the title and text of the i-th retrieved passages using `question` as the query.

> **SYSTEM MESSAGE:** You are a helpful assistant which answers a question based on a given set of documents. Please add reference (e.g. [0]) to the document in the answer.
>
> **USER MESSAGE:** Documents:
>
> [0] Paper title: `title_0`
>
> Passage: `passage_0`
>
> [1] Paper title: `title_1`
>
> Passage: `passage_1`
>
> [2] Paper title: `title_2`
>
> Passage: `passage_2`
>
> [3] Paper title: `title_3`
>
> Passage: `passage_3`
>
> [4] Paper title: `title_4`
>
> Passage: `passage_4`
>
> Question: `question`
> Answer:

Prompt for answer revision of information seeking instructions We first prompt the model to generate a summary s_t given the previous answer (a_{t-1}), the retrieved passages and the instruction i_t.

> **SYSTEM MESSAGE:** You are a helpful assistant which generates an intermediate answer given a feedback to the previous answer. Please add reference (e.g. [0]) to the document in the answer.
>
> **USER MESSAGE:** Documents:
>
> [0] Paper title: `title_0`
>
> Passage: `passage_0`
>
> [1] Paper title: `title_1`
>
> Passage: `passage_1`
>
> [2] Paper title: `title_2`
>
> Passage: `passage_2`
>
> [3] Paper title: `title_3`
>
> Passage: `passage_3`
>
> [4] Paper title: `title_4`

---

[11] https://www.gradio.app/

[12] https://platform.openai.com/docs/api-reference

[13] https://docs.together.ai/reference/inference

Passage: `passage_4`

Previous answer:a_{t-1}

Feedback:i_t

Generate an answer to the feedback.

We then prompt the model to generate an updated answer, given the original question, the answer in the previous turn a_{t-1}, the instruction i_t and the summary s_t. For `llama2` , we add an additional instruction at the end "Do not generate "Sure", directly generate the updated answer. Updated answer: ".

SYSTEM MESSAGE: You are a helpful assistant which generates an intermediate answer given a feedback to the previous answer. Please add reference (e.g. [0]) to the document in the answer.

USER MESSAGE: Original question: `question`

Previous answer: a_{t-1}

Feedback: i_t

Extra information related to the feedback: s_t

Answer the original question and incorporate the feedback to the previous answer. Preserve all relevant information from previous answer.

Prompt for answer revision of stylistic instructions We include the original question, answer in the previous turn (a_{t-1}) and the instruction i_t in the prompt. For `llama2` , we add an additional instruction at the end "Do not generate "Sure", directly generate the updated answer. Updated answer: ".

SYSTEM MESSAGE: You are a helpful assistant which answers a question based on a given set of documents. Please add reference (e.g. [0]) to the document in the answer.

USER MESSAGE: Original question: `question`

Previous answer: a_{t-1}

Feedback: i_t

Answer the original question and incorporate the feedback to the previous answer. Preserve all relevant information from previous answer.

Prompt length On average, the prompt for generating an answer revision for information seeking instructions (with question, previous answer, instruction and retrieved passages) consists of 607 words. The retrieved passages consists of 560 words on average. The prompt for answer revision for stylistic instructions (with question, previous answer and instruction) consists of 458 words on average.

A.2 Examples

Examples turns with good, `neutral` and bad rating can be found in Table 8. Example of questions annotated (§3.1) can be found in Table 7.

A.3 Implementation details for automatic instruction analysis with GPT-4

Manual analysis We randomly sample 50 interaction turns and manually group the instructions into fine-grained categories. We discover 8 fine-grained categories: requesting for more information (46%); expanding answer coverage (6%); asking for clarification (6%), categorization (6%), removal (10%), specific edits (2%), condense (12%) and reorganization (10%).

GPT-4 analysis We construct a prompt with definition and in-context examples for each of the category. The prompt we use to perform automatic instruction analysis in in Table 10. We prompt `gpt-4` with temperature of 0 and top_p=1. The distribution classified by GPT-4 is presented in 3. We see a slight distribution differences between the large-scale GPT-4 analysis and small-scale manual analysis, with GPT-4 identifying more instructions as "specific edits" and less as "categorization". We merge categories with fewer than 5% of examples into "Others".

More statistics We report additional statistics for each instruction type in Table 6. We filter out turns where the response is rated as `neutral` or bad with an edited answer, resulting in a total of 815 turns. Among the coarse-grained instruction types, we see that stylistic instructions are longer, yet with less edit ratio compared to information seeking ones. Specific edit instructions are the longest, while instructions asking for more information requires the most amount of edits to the previous answer (highest edit ratio).

A.4 Implementation details for error analysis

Manual analysis We randomly sample 10 interaction turns per model which are rated as `neutral` or `bad`, resulting in 30 total turns. We then group them into 5 fine-grained categories based on the question and comment.

GPT-4 analysis The prompt we use to perform automatic instruction analysis in in Table 11. We prompt `gpt-4` with temperature of 0 and top_p=1.

A.5 Implementation details for automatic evaluation of instruction following

A.5.1 GPT-4

We prompt `gpt-4-1106-preview` [14] for this task, whose context window can fit the ten-shot examples. We decode with temperature of 0 and top_p=1. Below are the prompt templates for the different baselines.

Zero-shot prompt template The model is given an instruction, the question, the answer in the previous turn a_{t-1}, the instruction i_t and the answer in the current turn a_t.

> **SYSTEM MESSAGE:** You are a helpful assistant in evaluating the quality of the outputs for a given instruction to update an answer for an question. Your goal is to score a given updated answer for the given instruction.
>
> **USER MESSAGE:** Score the updated answer for the given instruction by comparing it with the original answer.You should give one of the two ratings: good, or bad.
>
> Give a bad rating if the updated answer either (1) only partially followed the instruction (for example, the instruction specify an edit of a certain length or at a certain location but the updated answer didn't follow the constraint) or (2) introduced changes that are not mentioned in the instruction compared to the original answer or (3) provided a vague answer (e.g. without naming actual method) or (4) added the requested information but made the answer less coherent/correct.
>
> Only give a good rating if the answer COMPLETELY followed the instruction and didn't have ANY issues mentioned above. Your response should be ONLY the ratings.
>
> # Question:
>
> # `question`
>
> # Original answer:
>
> # a_{t-1}
>
> # Instruction:
>
> # i_t
>
> # Updated answer:
>
> # a_t
>
> # Rating of the Updated answer:
>
> #

Few-shot prompt template Each few-shot example j contains the question, the answer in the previous turn, the instruction, the answer in the current turn and the rating of the answer.

> **SYSTEM MESSAGE:** You are a helpful assistant in evaluating the quality of the outputs for a given instruction to update an answer for an question. Your goal is to score a given updated answer for the given instruction.
>
> **USER MESSAGE:** Score the updated answer for the given instruction by comparing it with the original answer.You should give one of the two ratings: good, or bad.
>
> Give a bad rating if the updated answer either (1) only partially followed the instruction (for example, the instruction specify an edit of a certain length or at a certain location but the updated answer didn't follow the constraint) or (2) introduced changes that are not mentioned in the instruction compared to the original answer or (3) provided a vague answer (e.g. without naming actual method) or (4) added the requested information but made the answer less coherent/correct.

---

[14] https://platform.openai.com/docs/models

Type	# words			Len ratio	Edit distance	Edit ratio
	inst	src	target			
More information	17	227	340	**1.60**	162	**0.85**
Expand coverage	39	295	398	1.48	195	0.78
Clarification	27	287	324	1.23	137	0.58
Remove	24	396	339	0.87	91.31	0.24
Specific edit	**75**	278	325	1.22	93	0.38
Condense	15	388	234	0.64	213	0.55
Reorganize	26	391	339	0.87	91.31	0.24
Others	23	327	342	1.36	212	0.88
Stylistic	**29**	361	316	0.94	141	0.42
Info seeking	22	259	355	**1.60**	176	**0.87**
GPT-4	24	260	286	1.28	145	**0.70**
GPT-3.5	23	289	332	**1.34**	145	0.68
Llama	**29**	379	387	1.20	173	0.56
Total	26	310	336	1.27	158	0.64

Table 6: We report statistics for each individual turns in our dataset: number of words in the instruction, number of words in the previous answer (src); number of words in the answer (target); the average length fraction between the source and the target; the token-level edit distance; and the fraction between th edit distance and the source text. We filter out turns where the response is rated as `neutral` or `bad` and there isn't an edited answer.

Only give a good rating if the answer COMPLETELY followed the instruction and didn't have ANY issues mentioned above. Your response should be ONLY the ratings.

# Question:

# `question_j`

# Original answer:

$a_{(t-1)j}$

# Instruction:

i_{tj}

# Updated answer:

a_{tj}

# Rating of the Updated answer:

r_{tj}

# Question:

# `question`

# Original answer:

a_{t-1}

# Instruction:

i_t

# Updated answer:

a_t

# Rating of the Updated answer:

#

A.5.2 T5

We finetune the model to output target sequence "Rating : r_i" with input sequence "Original question: q Previous answer: a_{t-1} Instruction: i_t Updated answer: a_t", where r_i is the traget binary rating, q is the question, a_{t-1} is the answer in the previous turn, i_t is the instruction and a_t is the answer in the current turn.

We use a batch size of 16 and an initial learning rate of `1e-4` with Adam optimizer and a linear learning rate schedule. We train the model for 5 epochs and choose the checkpoint with the best validation accuracy. The hyperparameters are manually searched by the authors.

A.6 Analysis on question generation with GPT-4

During the initial phase of the project, we explored an automatic version of the pipeline described in §3.1 by using GPT-4 to generate questions. We describe the pipelines below:

Step 1: Extracting and filtering related works We extract related work paragraphs from papers published in NLP venues (ACL, EMNLP, NAACL, etc.) using S2ORC (Lo et al., 2020). We extract the papers cited and filter out paragraphs with more than 30% of the papers missing from S2ORC or with less than 4 cited papers available. This gives us an initial set of (q, $\mathbf{D}$).

Step 2: Prompting LMs to generate questions For each related work paragraph, we prompt GPT-4 to generate five questions using the below prompt. We set the temperature to be 1 and top_p=1. `paper_title` is the title of the paper from which the related work paragraph is extracted.

> **SYSTEM MESSAGE:** You are a helpful assistant which generates five questions that the paragraph is addressing. The question should require multiple sentences to answer. Don't ask multiple sub questions in a single question. Don't refer to specific paper in the question.
> **USER MESSAGE:** Paper title: `paper_title`
> Passage: `related work paragraph`

Step 3: Question filtering We employ two filtering steps to filter out questions that do not fulfill the desiterata that we described in §3.1. First, we use a heuristic rule to filter out questions that contain keywords which make the question anchored to specific papers (e.g. “this paper”, “mentioned”, “author”). This process filters out 23% of the questions generated.

We then prompt GPT-4 on whether the generated question (`question`) can be answered by the related work paragraph using the following prompt. This process filters out 12% of the question which are rated as `No` or `Paratially` by GPT-4.

> **SYSTEM MESSAGE:** You are a helpful assistant.
> **USER MESSAGE:** Check if the paragraph answers the question. Reply Yes, No or Partially.
>
> Question: `question`
>
> Paragraph: `related work paragraph`

How are features of text and images fused in multimodal summarization models?
What are the different methods proposed for improving factual consistency in summaries generated by summarization systems?
Is it possible to extract the data used to train a language model and if so, how to do it?
How are pre-training corpora constructed for language models?
How are cross-lingual language models pre-trained?

Table 7: Example questions annotated (§3.1).

Manual examination We conduct a manual examination on the quality of the $(q, \mathbf{D})$ pairs generated through this automatic pipeline. We look at both the question and an initial answer generated by GPT-4 given $q, \mathbf{D}$, using the prompt described in §A.1. Two of the authors randomly sample 50 (question, initial answer) pairs to examine both the quality of the question and the initial answer.

We found that only 60% of the questions fulfill our desiderata described in §3.1. 28% of the question do not make sense to a researcher (e.g. “How do previous approaches to multi-task learning and domain adaptation try to create a universal representation space using encoders and architectures?”), 6% of the questions are too specific or niche (e.g. “What are the primary components used by the VGVAE model to represent the semantics and syntax of a sentence in a monolingual setting?”) and 6% of the questions do not stand alone (e.g. “How do labels like "other" or "information providing" act indicate issues with these models?”). For the questions that fulfill our desiderata, GPT-4 is unable to generate a good initial answer for 26% of them, often due to retrieval failure (e.g. “The documents provided do not discuss the potential issues with using word segmentation information and dependency trees for Chinese Named Entity Recognition (NER) as extra features.”). We note that this is not necessarily an issue with the retrieval system, but also an indication that the question could be irrelevant to the papers cited.

Overall, we find that this pipeline does not produce high quality (q, D) pairs needed for **KIWI**. Hence we proceed with human annotations as described in §3.1.

Rating	Example
`good`	**Original question:** Are there any datasets available fro table-to-text generation tasks in NLP? **Previous answer**: Yes, there are datasets available for table-to-text generation tasks in NLP. State-of-the-art models for data-to-text generation tasks have been trained and evaluated on various datasets. One commonly used dataset is the WebNLG dataset [0]. The DART model, for example, has achieved new state-of-the-art results on all WebNLG 2017 test set splits by fine-tuning T5-large on DART [0]. Other competitive models, both pretrained and not pretrained, have also been evaluated on this dataset [0] The DART dataset has been used for data augmentation and has shown to improve the performance of existing models on the WebNLG 2017 dataset [0]. The TOTTO dataset is another dataset used for table-to-text generation tasks [5]. However, state-of-the-art neural models have been found to struggle in generating faithful results on the TOTTO dataset, despite the high quality of the training data [5]. The performance gap between the models and the oracle upper-bound, which treats one of the references as a candidate, has been shown to be considerable [5]. The WikiTableT dataset focuses on generating Wikipedia article sections based on structured data [4]. However, the performance of state-of-the-art models on this dataset has not been explicitly mentioned in the provided documents. Overall, while state-of-the-art models have achieved impressive results on these datasets, there is still room for improvement and further research in the field of data-to-text generation [0] [4] [5]. **Instruction:** What metric was used to measure performance on these datasets? **Updated answer:** Yes, there are datasets available for table-to-text generation tasks in NLP. State-of-the-art models for data-to-text generation tasks have been trained and evaluated on various datasets. One commonly used dataset is the WebNLG dataset [0]. The participating systems on the WebNLG dataset were evaluated using three automatic metrics [1]. The DART model, for example, has achieved new state-of-the-art results on all WebNLG 2017 test set splits by fine-tuning T5-large on DART [0]. Other competitive models, both pretrained and not pretrained, have also been evaluated on this dataset [0]. The DART dataset has been used for data augmentation and has shown to improve the performance of existing models on the WebNLG 2017 dataset [0]. The TOTTO dataset is another dataset used for table-to-text generation tasks [5]. The evaluation on the TOTTO dataset used the BLEU metric, which measures n-gram overlap between the reference and the prediction at the corpus level [5]. However, state-of-the-art neural models have been found to struggle in generating faithful results on the TOTTO dataset, despite the high quality of the training data [5]. The performance gap between the models and the oracle upper-bound, which treats one of the references as a candidate, has been shown to be considerable [5]. The WikiTableT dataset focuses on generating Wikipedia article sections based on structured data [4]. The performance of state-of-the-art models on this dataset has not been explicitly mentioned in the provided documents. Overall, while state-of-the-art models have achieved impressive results on these datasets, their performance was measured using various metrics such as BLEU and REP [1] [4] [5]. These metrics provide quantitative measures to assess the performance of models on these datasets and compare different approaches. However, there is still room for improvement and further research in the field of data-to-text generation [0] [4] [5]. **Free-form comment:** The model provided the requested metrics.
`neutral`	**Original question:** What approaches have been proposed to edit a generated text to be factually consistent with a collection of evidence? **Previous answer:** In the context of factuality correction, a "collection of evidence" refers to a set of information or data that supports or refutes a claim or statement. This collection typically consists of retrieved snippets or passages from various sources, such as web pages or documents, which are used to assess the factual consistency of the generated text and make necessary revisions [1]. Approaches that have been proposed to edit a generated text to be factually consistent with a collection of evidence include: 1. Fact extraction or attention on fact triples: This approach involves extracting and incorporating factual information from the evidence collection into the generated text. Fact triples, which consist of subject-predicate-object statements, can be used to represent these extracted facts [5]. 2. Natural language inference or question-answering models for consistency checking: These models can be employed to verify the factual consistency of the generated text with the evidence collection. By comparing the generated text with the evidence, these models can identify any inconsistencies and suggest appropriate modifications [5]. 3. Training the model on artificial datasets: In order to improve factuality correction, the model can be trained on artificial datasets that contain both generated text and corresponding evidence. This approach helps the model learn to align the generated text with the evidence and make necessary revisions [5]. 4. Adversarial data generation process using Infilling Language Models: This approach involves generating adversarial examples by filling in missing information in the generated text. By iteratively generating and updating the text, the model can be fine-tuned to ensure factual consistency with the evidence [4]. 5. Factual error correction by editing inconsistent parts in generated summaries: This approach focuses on identifying and editing specific parts of the generated text that are inconsistent with the evidence. By revising these inconsistent parts, the overall factuality of the text can be improved [5]. 6. Factual error correction by rewriting claims to better align with retrieved evidence: This approach involves rewriting or modifying the claims made in the generated text to align them more closely with the evidence. By making these adjustments, the generated text can be made more factually consistent [6]. These approaches from papers [0], [2], [3], [4], [5], and [6] aim to enhance the factuality of generated texts by incorporating evidence and making necessary changes to ensure factual consistency. **Instruction:** Now keep the content and the order of the text as it is, but re-write the text into 1 paragraph only **Updated answer**: Approaches that have been proposed to edit a generated text to be factually consistent with a collection of evidence include: extracting and incorporating factual information from the evidence collection into the generated text using fact extraction or attention on fact triples [5]; employing natural language inference or question-answering models to verify the factual consistency and suggest modifications [5]; training the model on artificial datasets that contain both generated text and corresponding evidence to align the text with the evidence [5]; generating adversarial examples by filling in missing information in the text and fine-tuning the model for factual consistency [4]; identifying and editing specific parts of the text that are inconsistent with the evidence to improve factuality [5]; rewriting or modifying the claims made in the text to align them more closely with the evidence [6]. These approaches aim to enhance the factuality of generated texts by incorporating evidence and making necessary changes to ensure factual consistency [0] [2] [3] [4] [5] [6]. [1] **Free-form comment:** It removed a lot from the text, including the definition

Table 8: Example turns rated as `good`, `neutral`. The `bad` example is in Table 9.

Rating	Example
bad	**Original question:** Is it possible to extract the data used to train a language model and if so, how to do it? **Previous answer:** Yes, it is possible to extract the data used to train a language model through extraction attacks. These attacks involve recovering specific training examples from a language model by leveraging its output [0]. The goal is to reconstruct verbatim training examples rather than just representative or "fuzzy" examples [0]. The process of performing extraction attacks typically involves several steps. First, the attacker generates probing queries designed to elicit specific responses from the language model [0]. These queries are carefully crafted to contain clues about the training data. Then, the attacker analyzes the responses generated by the language model to identify potential matches with the training data [0]. By observing patterns and similarities in the modelś output, the attacker can infer the presence of specific training examples. The attacker iteratively refines their probing queries based on the model responses, gradually narrowing down the potential training examples [0]. This iterative refinement process involves generating new queries that are tailored to elicit more precise responses from the language model. Once the attacker has identified potential training examples, they can reconstruct the verbatim content by combining multiple queries and responses [0]. By carefully analyzing the modelś behavior, the attacker can piece together the training data that the language model has memorized. Extraction attacks have been shown to be practical and efficient, allowing adversaries to extract hundreds of training examples from a language model [0][5][7]. These attacks have been successfully performed against various language models, including GPT-2 and clinical language models like ClinicalBERT [5][7]. Furthermore, extraction attacks can succeed even when a specific training example appears only a few times in the data [0]. In addition to extraction attacks, membership inference attacks can also be used to extract training data from language models. These attacks focus on determining whether a given example is part of the training data [5]. Adversaries train shadow models that mimic the behavior of the attacked model and use them to train an auditing model that can predict membership in the training dataset [5]. Membership inference attacks have been performed on different NLP models, including masked language models like GPT-2 and clinical language models such as ClinicalBERT, successfully extracting training data from these models [5][7]. Therefore, both extraction attacks and membership inference attacks can be used to extract the data used to train a language model [0][5][7]. **Instruction:** Combine the second and third paragraphs into one paragraph of no more than five sentences in which the steps to perform an extraction attack are listed. **Upadted answer:** Yes, it is possible to extract the data used to train a language model through extraction attacks. These attacks involve recovering specific training examples from a language model by leveraging its output [0]. The process of performing extraction attacks typically involves several steps. First, the attacker generates probing queries designed to elicit specific responses from the language model [0]. These queries are carefully crafted to contain clues about the training data. Then, the attacker analyzes the responses generated by the language model to identify potential matches with the training data [0]. By observing patterns and similarities in the model's output, the attacker can infer the presence of specific training examples. The attacker iteratively refines their probing queries based on the model responses, gradually narrowing down the potential training examples [0]. This iterative refinement process involves generating new queries that are tailored to elicit more precise responses from the language model. Once the attacker has identified potential training examples, they can reconstruct the verbatim content by combining multiple queries and responses [0]. By carefully analyzing the model's behavior, the attacker can piece together the training data that the language model has memorized. Extraction attacks have been shown to be practical and efficient, allowing adversaries to extract hundreds of training examples from a language model [0][5][7]. These attacks have been successfully performed against various language models, including GPT-2 and clinical language models like ClinicalBERT [5][7]. Furthermore, extraction attacks can succeed even when a specific training example appears only a few times in the data [0]. In addition to extraction attacks, membership inference attacks can also be used to extract training data from language models. These attacks focus on determining whether a given example is part of the training data [5]. Adversaries train shadow models that mimic the behavior of the attacked model and use them to train an auditing model that can predict membership in the training dataset [5]. Membership inference attacks have been performed on different NLP models, including masked language models like GPT-2 and clinical language models such as ClinicalBERT, successfully extracting training data from these models [5][7]. Therefore, both extraction attacks and membership inference attacks can be used to extract the data used to train a language model [0][5][7]. [0] **Free-form comment:** The model did not shorten the answer as requested.

Table 9: Example turns rated as bad by the annotator.

You are given an instruction to improve an answer to a question. Your job is to assign the instruction into one of the category below.

Example 1
# Original question: Are there any datasets available fro table-to-text generation tasks in NLP? # Instruction: How well do state-of-the-art models perform on these datasets?
# Category: asking for more information
# Explanation: This category of instruction ask for more information related to the question (e.g. example, limitation, evaluation results, background information, citations, definition).

Example 2
# Original question: How has prior work study and evaluate robustness of multimodal models?
# Instruction: As first sentence of the text, add a very short one sentence definition of what are "multimodal models"
# Category: asking for more information
# Explanation: This category of instruction ask for more information related to the question (e.g. example, limitation, evaluation results, background information, citations, definition).

Example 3
# Original question: How did prior work explore learning from human feedback for different NLP tasks?
# Instruction: Try to add very short information also from papers 0,1,3,5,6,7,8. Keep the rest of the text exactly as it is.
# Category: expand answer coverage
# Explanation: This category of instruction asks for more information directly answering the question, usually from multiple papers.

Example 4
# Original question: What are the existing approaches for the lip-reading task?
# Instruction: You mention that some methods use a network to predict phoneme probabilities and were trained using CTC loss and that it was trained using YouTube videos. Does this mean that such methods require phoneme level labels for the data to be trained, or does word level transcriptions work?
# Category: asking for clarification
# Explanation: This category of instruction asks for clarification of a concept already in the answer.

Example 5
# Original question: What are the prior efforts that develop RL environments with language-informed tasks?
# Instruction: Organize the papers better in categories of what aspect of language-conditioned RL tasks they are tackling.
# Category: categorization
# Explanation: This category of instruction requests to categorize the content in the answer.

Example 6
# Original question: How can information stored in knowledge bases be injected into language models task-specific model fine-tuning?
# Instruction: Please shorten the answer to a maximum of 6 paragraphs.
# Category: condense
# Explanation: This category of instruction requests to shorten/summarize the answer.

Example 7
# Original question: How can I quantize the weights of a generative language model and still achieve reasonable model performance?
# Instruction: Without changing anything else, delete every instance of the string "(Feedback)", and delete the string "[1] “Compression of Generative Pre-trained Language Models via Quantization”" at the end of the answer.
# Category: remove
# Explanation: This category of instruction requests to remove/delete certain part of the answer.

Example 8
# Original question: What techniques have been proposed to measure and evaluate the effectiveness of task-oriented dialogue systems?
# Instruction: Move the paragraph starting with "Finally, there are also various task-specific evaluation metrics" to after the discussion of user satisfaction modeling, and make sure to cite paper [2] in that paragraph.
# Category: reorganize
# Explanation: This category of instruction requests to reorganize the answer(e.g. move things around, etc.).

Example 9
# Original question: What methods have been proposed to categorize the kinds of editing operations that occur during text simplification?
# Instruction: Immediately after this sentence: "In the process of text simplification, different text-editing operations are used. " add this sentence "The editing operations that occur during text simplification can be usually categorized after doing literature review and reading previous publications on text simplification."
# Category: direct verbatim edit
# Explanation: This category of instruction requests to directly add information that are provided in the instruction verbatim to the answer.

Example 10
# Original question: How have contrastive learning techniques been applied to learn dense sentence representations?
# Instruction: paragraph 3 does not describe another approach,
# Category: others
# Explanation: This category of instruction doesn’t belong to any of the category above.

Now, assign a category for the example below. ONLY ASSIGN ONE OF THE CATEGORY ABOVE. DONOT INVENT YOUR OWN CATEGORY. DON’T PROVIDE AN EXPLANATION.

Table 10: Prompt for automatic instruction analysis.

You are given an instruction for a model to improve an answer to a question and a comment on how the updated answer followed the instruction. Your job is to assign the comment into one of the category below.

Example 1
# Instruction: Now add this information: ""[2] introduce a controllable summarization model that provides a mechanism for users to specify high level attributes such as length, style, or entities of interest. This enables personalized generation.""
# Comment: It added the information, but in the wrong place, so it looks as an extension of another method. # Category: added information but made answer less coherent/correct
# Explanation: This category of comment will mention that the instruction is followed, but the updated answer is less coherent (for example, become too long; has a weird structure, or is not correct).
In the example, the comment mentioned that the updated answer has the information added but the updated answer is confusing.

Example 2
# Instruction: Remove the tenth sentence.
# Comment: The sentence was not removed as requested.
# Category: instruction is ignored
# Explanation: This category of comment will mention that the instruction is not followed at all. In this example, the comment mentioned that requested change was not performed.

Example 3
# Instruction: As a next sentence after this one "These parameters include the weights and biases of the model's layers, such as the embedding layer, encoder, decoder, and attention mechanisms [0]." add a short sentence, which explains why parameters sharing is necessary in multilingual models used for machine translation
# Comment: It added an explanation, but deleted a whole paragraph
# Category: introduced unrequested changes
# Explanation: This category of comment will mention that the instruction is followed but also there are unrequested changes performed. In this example, the comment mentioned that the requested change (explanation) was incorporated but it also introduced unrequested change (a whole paragraph was deleted).

Example 4
# Instruction: Great answer! Without changing anything else, in the last paragraph, mention some tasks that benefit from using explanations, and other tasks where it impedes performance.
# Comment: The answer is overall still good but the new material is too long, and the model disregarded the part of the prompt where it was asked to put this material in the last paragraph. I will try to re-prompt.
# Category: failed to follow hard constraint
# Explanation: This category of comment will mention one ore more hard constraint in the instruction is not followed by the model.
The hard constraint can be a specific location (at the beginning of the answer), length of the added/final answer (a very short definition), or an action to avoid ("do not...").
In this example, the comment mentioned that the updated answer didn't follow the constraint in the instruction (which mention the added information should be in the last paragraph).

Example 5
# Instruction: I notice a lot of redundant sentences, such as repeated mention of something like "compress the model while maintaining performance." Can you make your answer more concise?
# Comment: model mostly just added paragraphs together without cutting actual content
# Category: others
# Explanation: This category of comment will mention that the model tried to follow the instruction (thus not completely ignore the instruction) but the updated answer is not satisfactory.
The updated answer doesn't has issue mentioned above (unrequested change or failed to follow hard constraint), but just in general has subpar quality.

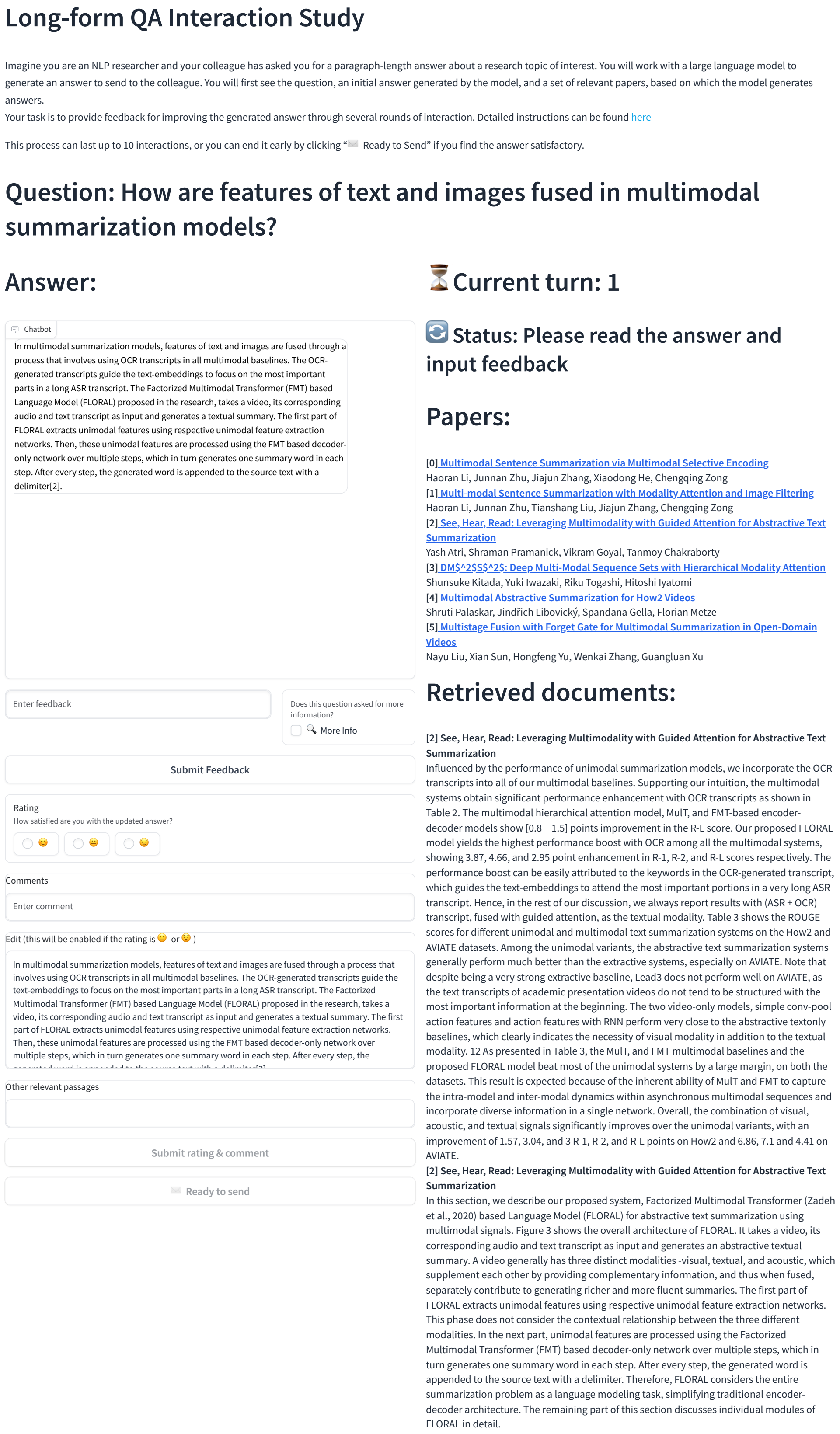
Now, assign a category for the example below. ONLY ASSIGN ONE OF THE CATEGORY ABOVE. DONOT INVENT YOUR OWN CATEGORY. DON'T PROVIDE AN EXPLANATION.

Table 11: Prompt for automatic comment analysis.

```
Require: Language model $M$, Retrieval model $R$, User $U$
Input: question $q$ and a set of relevant papers $D$
Output: a series of interaction $\mathcal{I}$, each consists of (instruction $i$, answer $a$, rating $r$, comment $c$, edited answer $a'$)

$\mathcal{I} \leftarrow \emptyset$
$a_0 = M(q, R(q, D))$ {Generate an initial answer}
$a_p \leftarrow a_0$ {Set initial answer as previous answer}
for $t \in \{1, \ldots, 10\}$ do
    $i_t, I_t = U(q, a_0, ...a_{t-1}, D)$ {User issues an instruction $i_t$, and $I_t$, the type of the instruction.}
    $p_t \leftarrow \emptyset$
    if $I_t ==$ info then
        $p_t = R(i_t, D)$
    end if
    $a_t = M(q, a_p, p_t, i_t)$
    $r_t, c_t = U(q, i_t, prev\_ans, a_t)$ {User provides a rating and comment for the updated answer.}
    $a'_t \leftarrow a_t$
    if $r_t \in \{$neutral, bad$\}$ then
        $a'_t = U(q, i_t, a_{t-1})$ {User optionally edits the updated answer.}
    end if
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{(i_t, a_t, r_t, c_t, a'_t)\}$
    if $U(q, a'_t) ==$ Good then
        break {Interaction ends if the user is satisfy with the answer at this turn.}
    else if $r_t! =$ Bad or $a_t! = a'_t$ then
        $a_p \leftarrow a'_t$ {Update previous answer if answer in this turn is not rated as bad OR edited by user.}
    end if
end for
```

Figure 3: We present the workflow for collecting **`KIWI`**. Detailed description for each step is in §2.



Long-form QA Interaction Study

Imagine you are an NLP researcher and your colleague has asked you for a paragraph-length answer about a research topic of interest. You will work with a large language model to generate an answer to send to the colleague. You will first see the question, an initial answer generated by the model, and a set of relevant papers, based on which the model generates answers.
Your task is to provide feedback for improving the generated answer through several rounds of interaction. Detailed instructions can be found here

This process can last up to 10 interactions, or you can end it early by clicking "✉️ Ready to Send" if you find the answer satisfactory.

Question: How are features of text and images fused in multimodal summarization models?

Answer:

In multimodal summarization models, features of text and images are fused through a process that involves using OCR transcripts in all multimodal baselines. The OCR-generated transcripts guide the text-embeddings to focus on the most important parts in a long ASR transcript. The Factorized Multimodal Transformer (FMT) based Language Model (FLORAL) proposed in the research, takes a video, its corresponding audio and text transcript as input and generates a textual summary. The first part of FLORAL extracts unimodal features using respective unimodal feature extraction networks. Then, these unimodal features are processed using the FMT based decoder-only network over multiple steps, which in turn generates one summary word in each step. After every step, the generated word is appended to the source text with a delimiter[2].

Enter feedback
Does this question asked for more information?
☐ 🔍 More Info
Submit Feedback
Rating
How satisfied are you with the updated answer?
○ 😊 ○ 😐 ○ 😟
Comments
Enter comment
Edit (this will be enabled if the rating is 😐 or 😟)

In multimodal summarization models, features of text and images are fused through a process that involves using OCR transcripts in all multimodal baselines. The OCR-generated transcripts guide the text-embeddings to focus on the most important parts in a long ASR transcript. The Factorized Multimodal Transformer (FMT) based Language Model (FLORAL) proposed in the research, takes a video, its corresponding audio and text transcript as input and generates a textual summary. The first part of FLORAL extracts unimodal features using respective unimodal feature extraction networks. Then, these unimodal features are processed using the FMT based decoder-only network over multiple steps, which in turn generates one summary word in each step. After every step, the

⏳Current turn: 1

🔄 Status: Please read the answer and input feedback

Papers:

[0] Multimodal Sentence Summarization via Multimodal Selective Encoding
Haoran Li, Junnan Zhu, Jiajun Zhang, Xiaodong He, Chengqing Zong
[1] Multi-modal Sentence Summarization with Modality Attention and Image Filtering
Haoran Li, Junnan Zhu, Tianshang Liu, Jiajun Zhang, Chengqing Zong
[2] See, Hear, Read: Leveraging Multimodality with Guided Attention for Abstractive Text Summarization
Yash Atri, Shraman Pramanick, Vikram Goyal, Tanmoy Chakraborty
[3] DM$^2$S$^2$: Deep Multi-Modal Sequence Sets with Hierarchical Modality Attention
Shunsuke Kitada, Yuki Iwazaki, Riku Togashi, Hitoshi Iyatomi
[4] Multimodal Abstractive Summarization for How2 Videos
Shruti Palaskar, Jindřich Libovický, Spandana Gella, Florian Metze
[5] Multistage Fusion with Forget Gate for Multimodal Summarization in Open-Domain Videos
Nayu Liu, Xian Sun, Hongfeng Yu, Wenkai Zhang, Guangluan Xu

Retrieved documents:

[2] See, Hear, Read: Leveraging Multimodality with Guided Attention for Abstractive Text Summarization
Influenced by the performance of unimodal summarization models, we incorporate the OCR transcripts into all of our multimodal baselines. Supporting our intuition, the multimodal systems obtain significant performance enhancement with OCR transcripts as shown in Table 2. The multimodal hierarchical attention model, MulT, and FMT-based encoder-decoder models show [0.8 – 1.5] points improvement in the R-L score. Our proposed FLORAL model yields the highest performance boost with OCR among all the multimodal systems, showing 3.87, 4.66, and 2.95 point enhancement in R-1, R-2, and R-L scores respectively. The performance boost can be easily attributed to the keywords in the OCR-generated transcript, which guides the text-embeddings to attend the most important portions in a very long ASR transcript. Hence, in the rest of our discussion, we always report results with (ASR + OCR) transcript, fused with guided attention, as the textual modality. Table 3 shows the ROUGE scores for different unimodal and multimodal text summarization systems on the How2 and AVIATE datasets. Among the unimodal variants, the abstractive text summarization systems generally perform much better than the extractive systems, especially on AVIATE. Note that despite being a very strong extractive baseline, Lead3 does not perform well on AVIATE, as the text transcripts of academic presentation videos do not tend to be structured with the most important information at the beginning. The two video-only models, simple conv-pool action features and action features with RNN perform very close to the abstractive textonly baselines, which clearly indicates the necessity of visual modality in addition to the textual modality. 12 As presented in Table 3, the MulT, and FMT multimodal baselines and the proposed FLORAL model beat most of the unimodal systems by a large margin, on both the datasets. This result is expected because of the inherent ability of MulT and FMT to capture the intra-model and inter-modal dynamics within asynchronous multimodal sequences and incorporate diverse information in a single network. Overall, the combination of visual, acoustic, and textual signals significantly improves over the unimodal variants, with an improvement of 1.57, 3.04, and 3 R-1, R-2, and R-L points on How2 and 6.86, 7.1 and 4.41 on AVIATE.
[2] See, Hear, Read: Leveraging Multimodality with Guided Attention for Abstractive Text Summarization
In this section, we describe our proposed system, Factorized Multimodal Transformer (Zadeh et al., 2020) based Language Model (FLORAL) for abstractive text summarization using multimodal signals. Figure 3 shows the overall architecture of FLORAL. It takes a video, its corresponding audio and text transcript as input and generates an abstractive textual summary. A video generally has three distinct modalities -visual, textual, and acoustic, which supplement each other by providing complementary information, and thus when fused, separately contribute to generating richer and more fluent summaries. The first part of FLORAL extracts unimodal features using respective unimodal feature extraction networks. This phase does not consider the contextual relationship between the three different modalities. In the next part, unimodal features are processed using the Factorized Multimodal Transformer (FMT) based decoder-only network over multiple steps, which in turn generates one summary word in each step. After every step, the generated word is appended to the source text with a delimiter. Therefore, FLORAL considers the entire summarization problem as a language modeling task, simplifying traditional encoder-decoder architecture. The remaining part of this section discusses individual modules of FLORAL in detail.

Figure 4: A screenshot of our annotation interface.