Personalized Abstractive Summarization by Tri-agent Generation Pipeline

Wen Xiao[‡] **Yujia Xie**[‡]
Giuseppe Carenini[†] **Pengcheng He**[‡]
[†]University of British Columbia, Vancouver, Canada
[‡] Microsoft Azure AI
{carenini}@cs.ubc.ca,
{wxiao,yujiaxie,penhe}@microsoft.com

Abstract

Tailoring outputs from large language models, like ChatGPT, to implicit user preferences remains a challenge despite their impressive generative capabilities. In this paper, we propose a tri-agent generation pipeline comprising a generator, an instructor, and an editor to enhance output personalization. The generator produces an initial output, the instructor automatically generates editing instructions based on user preferences, and the editor refines the output to align with those preferences. The inference-only large language model (ChatGPT) serves as both the generator and editor, with a smaller model acting as the instructor to guide output generation. We train the instructor using editor-steered reinforcement learning, leveraging feedback from a large-scale editor model to optimize instruction generation. Experimental results on two abstractive summarization datasets demonstrate the effectiveness of our approach in generating outputs that better meet user expectations. [1]

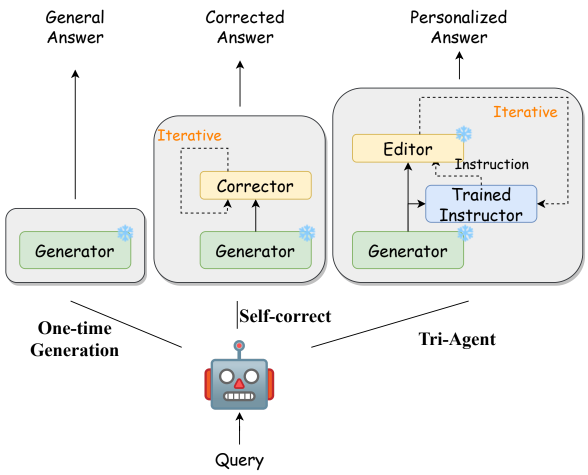

Figure 1: Comparison between different generation paradigms. The left one is the general one-time generation process, the middle one is from Welleck et al. (2022), which uses a trained corrector to make corrections on the generated text, usually dealing with specific issues, like eliminating hallucination or toxicity, and the right one is the proposed tri-agent pipeline.

1 Introduction

Large language models, exemplified by prominent models such as InstructGPT (Ouyang et al., 2022) and ChatGPT[2], have emerged as essential resources in the field of natural language processing (NLP). These models have shown an extraordinary level of proficiency across a broad spectrum of NLP tasks, including machine translation, question answering, and text summarization. In light of their potential to drive further innovation in language-based technologies, the research community has exhibited growing enthusiasm for exploring and advancing large language models. However, despite the impressive generation quality achieved by these models, a persistent challenge lies in tailoring their outputs to meet user's preference (Liu et al., 2022b). In several scenarios, it has been observed that the outputs of language models do not consistently satisfy users' preferences or expectations (Bubeck et al., 2023). A prevalent approach to addressing this limitation involves the careful crafting of prompts to steer the models in producing outputs that better align with users' objectives. Nonetheless, as noted in existing research (Reid and Neubig, 2022), the conventional one-time left-to-right generation process of language models contrasts with the iterative refinement and editing approach commonly employed by humans. Furthermore, prior works (Gu et al., 2019; Reid and Zhong, 2021) have demonstrated the efficacy of the generate-and-edit process compared to one-time generation, even with a single editing iteration. Motivated by these findings, this paper explores the integration of large language models (ChatGPT) into an automatic iterative editing pipeline.

[1]Code is available at https://github.com/Wendy-Xiao/chatgpt_editing_summ
[2]https://openai.com/blog/chatgpt

Findings of the Association for Computational Linguistics: EACL 2024, pages 570–581
March 17-22, 2024

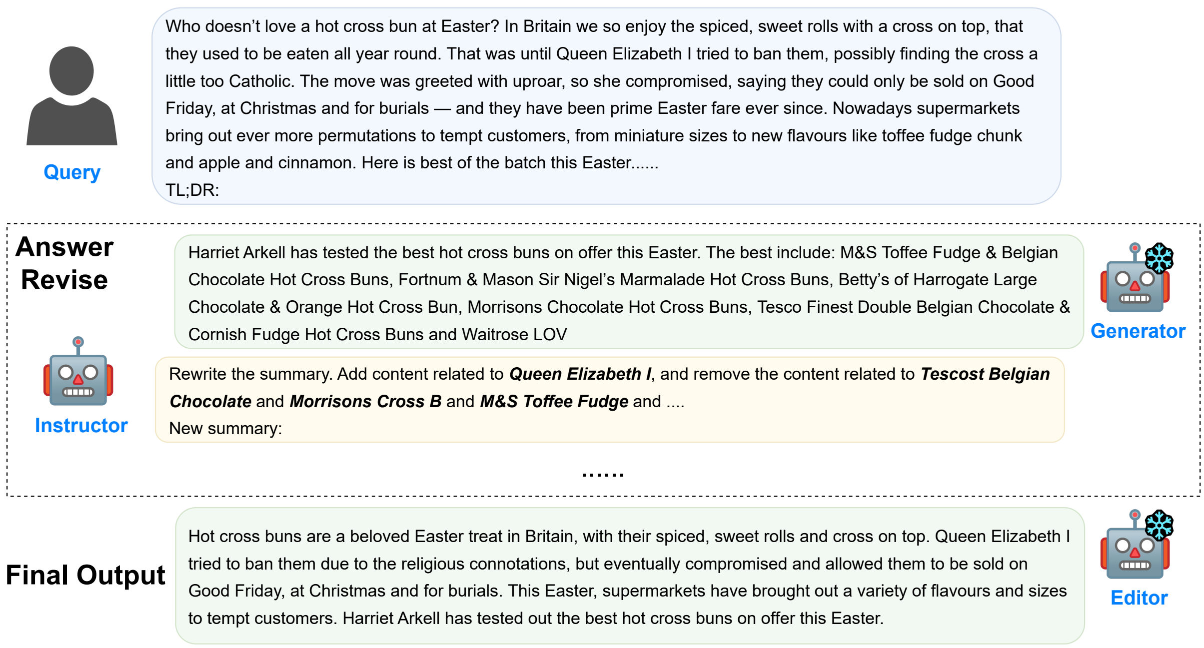

Figure 2: An illustration of the proposed tri-agent generation pipeline. When a query is given, the generator first generates an initial answer, and the instructor provide an instruction on how to make the answer more tailored to user's preference, and finally the editor generates a personalized answer with the given instruction.

In contrast to the approach taken by Welleck et al. (2022), where the generation process is decomposed into a generator and a corrector, our methodology involves a three-component decomposition consisting of a **generator, instructor, and editor** (refer to Figure 1). This structure allows us to leverage inference-only large models for the complex tasks of content generation and correction, while utilizing smaller models for the simpler task of generating user-specific editing instructions. The instructor is designed to provide targeted directives for editing and refining the initial outputs of the generator. It is initialized by training on human-authored, or oracle, instructions, which can be obtained by the history of user's behaviour. Following this, the model is then fine-tuned through editor-steered reinforcement learning, wherein the reward function directly quantifies the degree to which **the edited output by the editor** align with user preferences, which enhances the model's compatibility with the editor.

We choose text summarization as the focal task for evaluating this novel framework, which is to generate concise and informative summary for the given document(s). In this paper, we conduct experimental evaluations on two summarization datasets (DeFacto (Liu et al., 2022b) and CNNDM (Nallapati et al., 2016)), focusing on user preference related to factual consistency and coverage. We employ ChatGPT as the generator and the editor model. Our experiments indicate that with the instructions generated by the small instructor model, the edited output is better aligned with user's preference on both datasets. Further experiments on the iterative editing shows that the output can better meet user's needs with more iterations of editing.

2 Overall Pipeline

In an effort to enhance the flexibility of the generation pipeline and optimize its compatibility with powerful large language models, we propose a novel decomposition of the generation process into three distinct components, as illustrated in Figure 2. These components include: (1) a **generator**, responsible for producing the initial output; (2) an **instructor**, tasked with generating natural language instructions that guide the editing of the initial output toward the direction of user preference; and (3) an **editor**, which refines the initial output in accordance with the provided instructions.

Since it has been demonstrated that large language models can act as both a generator and an editor model, we have chosen to utilize an inference-only large language model, specifically ChatGPT, as our generator and editor. While it is possible to further fine-tune these large language models

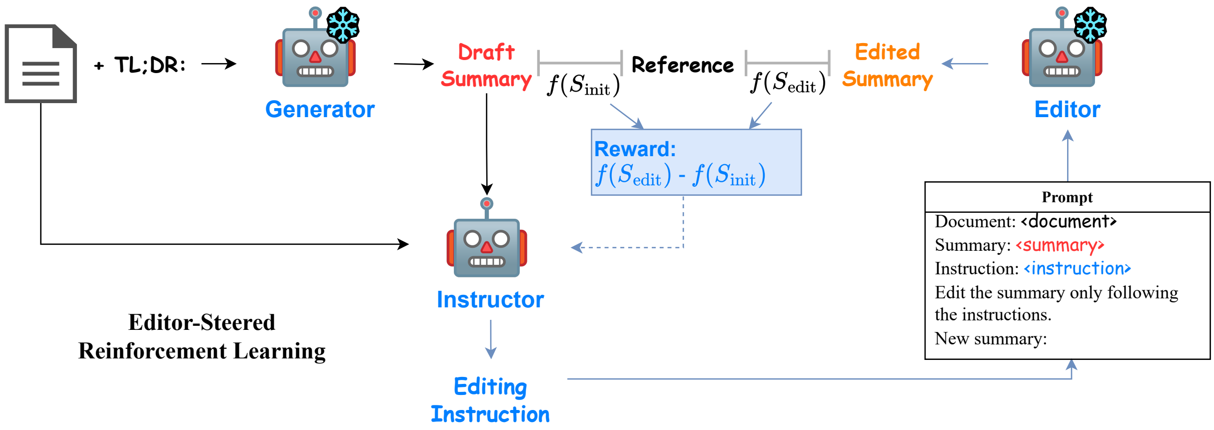

Figure 3: Editor-steered Reinforcement Learning for the instructor. We fine-tune the instructor using editor-steered reinforcement learning to maximize the expected performance of the editor (e.g., ChatGPT).

to serve as instructors, practical limitations such as computational resources (Touvron et al., 2023) and access restrictions (Ouyang et al., 2022) may prevent direct fine-tuning, as has been done in previous works (Welleck et al., 2022; Liu et al., 2022a). Therefore we propose to train a smaller model with editor-steered reinforcement learning to function as a user-specific instructor (as introduced in Section 3), which guides the editor in revising the initial output to achieve better alignment with human expectations.

3 Editor-steered Instructor

As introduced above, the central objective of the proposed instructor is to produce precise and actionable instructions that can guide a large language model in correcting the original summary to align more closely with the user's preference. To achieve this, we employ a two-phase training process that is designed to enable the instructor to work synergistically with large language models.

Specifically, given the document D, an initial summary, denoted as S_{init}, is generated using a generator (either a summarization model or a large language model). The objective of the instructor is to take D and S_{init} as inputs and generate a set of instructions $I = \{i_1, i_2, ..., i_k\}$, aiming to guide the editor model in generating an edited summary that is more closely aligned with the user's preference. Finally, the editor takes D, S_{init}, and I as input and generates a revised summary S_{edit} according to the given instructions.

3.1 Step 1: Supervised Learning

During the initial training phase, we generate a set of oracle instructions tailored to the user's historical preferences for summary correction.[3] These oracle instructions serve as ideal examples of the instructions that our instructor should produce. We then train the instructor model in a supervised manner, with negative log likelihood loss, i.e.,

$$L = \sum_{k} P(i_1, i_2, ..., i_k | D, S_{\text{init}}).$$

The goal of this phase is to establish a solid foundation for the instructor to generate instructions that align with user expectations, by enabling it to learn the relationship between the input (source documents and initial summaries) and the desired output (oracle instructions).

3.2 Step 2: Editor-steered Reinforcement Learning

In the second phase, we further fine-tune the instructor model using editor-steered reinforcement learning techniques (see Figure 3), specifically using the NLPO algorithm (Ramamurthy et al., 2023).

A key aspect of this phase is the design of the reward function, which serves as the guiding signal for the RL-based fine-tuning process. To ensure that the generated instructions are compatible

[3]These oracle instructions are constructed by simulating the user's preferences using human-written summaries as references, which reflect the distinct summarization preferences of each source. For instance, CNN and DailyMail may exhibit specific tendencies in the summaries it generates for news articles.

with the editor model and lead to meaningful summary corrections, the reward function is formulated based on the edited summary, which is generated by the editor model using prompts that include the source documents, initial summaries, and editing instructions provided by the instructor model (see the example prompt shown at right-bottom of Figure 3).

To quantify the quality of the edited summary, we employ a scoring function $f(\cdot)$ that measures the extent to which the summary fulfills the user's preference. As we focus on the coverage and factual consistency of the generated summaries as the user's requirements, the scoring function $f(\cdot)$ is then set as the sum of ROUGE score and knowledge coverage, which measures the similarity of the entity level coverage with the reference summaries,

$$f(S) = \alpha \text{ROUGE}(S, S_{\text{ref}}) + \beta Cov(S, S_{\text{ref}}).$$

The reward signal itself is defined as the difference in scores between the initial and edited summary, which is designed to capture improvements in summary quality, with higher rewards corresponding to more substantial improvements,

$$Reward = f(S_{\text{edit}}) - f(S_{\text{init}}).$$

This phase aims to enhance the model's ability to generate instructions that not only adhere to user requirements, but also effectively guide the large language model to produce improved summaries.

4 Experiments

We conduct experiments on two distinct datasets, each capturing different facets of user preferences.

4.1 Scenario 1: Factual Consistency on DeFacto

In the initial experimental scenario, we opt to emphasize factual consistency as the primary criterion for users' summary preferences.[4] We employ the DeFacto dataset (Liu et al., 2022b), a resource specifically curated to enhance the factual consistency of machine-generated summaries through the inclusion of human-annotated demonstrations and feedback. The dataset consists of $701/341/779$ data examples in train/validation/test set respectively.[5] Each data entry in the DeFacto dataset comprises a source document and an initial summary generated by PEGASUS (Zhang et al., 2020). Annotators are tasked with providing an instruction that guides the modification of the initial summary to enhance factual consistency. Additionally, annotators generate a revised summary that adheres to the provided instructions and exhibits improved factual consistency.

To evaluate the alignment between the system-generated instructions and the human-written instructions, we employ the ROUGE score as our evaluation metric. Additionally, we assess the quality of the generated summaries with respect to human expectations and factual accuracy using a combination of metrics, including ROUGE scores and factualness scores. Specifically, we utilize the DAE (Dependency Arc Entailment) metric (Goyal and Durrett, 2021) and the QFE (Question-answering for Factual Evaluation) metric (Fabbri et al., 2022) to quantify the factualness of the generated summaries. These metrics provide a comprehensive assessment of summary quality in terms of both alignment with human expectations and adherence to factual correctness.

Settings We use FlanT5-large (700M) (Chung et al., 2022) as the backbone model for the instructor. The training process for the instructor is executed in two phases, as detailed in Section 3.

Results First of all, we assess the potential of ChatGPT to serve as an editor model, capable of revising summaries in accordance with human-provided instructions. The results of this assessment, presented in Table 1, indicate that ChatGPT performs comparably to a supervised model when supplied with source documents, initial summaries, and human-written editing instructions as input, as demonstrated by comparable ROUGE scores and factualness scores. These findings affirm that ChatGPT is effective as a summary editor when appropriate editing instructions are provided.

Then, we evaluate the system-generated instructions in comparison to human-authored instructions. Our objective is to determine the extent to which ChatGPT and trained instructors can accurately discern user requirements and subsequently produce corresponding instructions. The results

[4]While factual consistency may serve as a typical criterion for summarizers in general, we leverage the instructor to acquire the ability to craft specific instructions that enhance the factual consistency of the summaries.

[5]Following the original paper, all the experiments are conducted on the examples labeled with errors.

Editor	DAE	QFE	R1	R2	RL
Initial Summary	0.699	1.837	76.03	66.34	74.11
Human Editor	0.906	2.717	100	100	100
T0PP-D+S+I (Sup)	0.904	2.470	88.74	83.16	87.48
ChatGPT (10-shot)	0.884	2.568	88.48	81.41	86.17

Table 1: The ROUGE score and factual consistency scores of edited summaries with human-written instructions on DeFacto, in comparison with the human-edited summaries. T0PP-D+S+I (Sup) is a supervised model with the source Documents, initial Summary and Instruction as the input (Liu et al., 2022b).

Model	R1	R2	RL
ChatGPT (Zero-shot)	36.05	22.98	30.66
ChatGPT (10-shot)	37.35	24.94	32.94
FlanT5 (Sup)	49.04	34.37	47.07
FlanT5 (RL)	48.05	32.94	46.23

Table 2: ROUGE score between generated instructions and human-written instructions on DeFacto.

of this evaluation are presented in Table 2. Notably, we observe that the instructions generated by ChatGPT do not effectively match human-written instructions, as evidenced by suboptimal performance in both zero-shot and few-shot settings. Although the instructor model we used is much smaller than ChatGPT (700M v.s. 175B), it shows the ability to generate instructions better aligned with the user's needs.

In the final set of experiments, presented in Table 3, we evaluate the performance of the editing model (ChatGPT) with the trained and RL fine-tuned instructors, as well as the instructions generated by ChatGPT in few-shot settings. The results demonstrate that summaries edited by ChatGPT, when utilizing a 10-shot prompt and instructions from the trained instructor, exhibit large improvements in factualness(as measured by DAE/QFE) compared to the original summaries . The implementation of reinforcement learning, incorporating ChatGPT-derived rewards, leads to additional enhancements in summary quality. Furthermore, we conduct experiments utilizing instructions generated by ChatGPT. While these instructions demonstrate suboptimal alignment with human-authored instructions, they yield unexpectedly high scores in terms of factualness, particularly as measured by the QFE metric. However, a notable decrease in ROUGE scores is observed in comparison to other methods. These findings suggest that ChatGPT possesses the capacity to generate instructions that target a specific and well-defined aspect (e.g., addressing factual inconsistencies), but may struggle to accurately discern and fulfill broader human expectations.

4.2 Scenario 2: Coverage on CNNDM

ChatGPT has demonstrated its capacity to produce fluent and informative summaries of news articles (Goyal et al., 2022). Despite its proficiency in generating coherent summaries, it may not always achieve the desired coverage of key topics, as expected by the user. In response to this challenge, we conduct an experiment to train and evaluate an instructor model specifically designed to guide the editing of summaries for improved knowledge coverage based on user's history. The instructor predicts the keywords to be added to or removed from the current summary, thereby providing actionable instructions to align the summary more closely with user preference. In practice, we assess knowledge coverage based on the extent to which the generated summaries match reference summaries in terms of keyword content.

We employ the CNNDM dataset (Nallapati et al., 2016) as our benchmark for this experiment, which contains pairs of articles and reference summaries, with the original reference summary serving as the target representation of user preference on the coverage. We acknowledge that, according to recent studies (Goyal et al., 2022), the reference summaries in the CNNDM dataset may exhibit some quality limitations, such as poor coherence. However, our primary focus in this experiment is on knowledge coverage rather than summary quality. We are interested in assessing the extent to which the generated summaries capture the key entities in the reference.

To measure knowledge coverage, we introduce an entity-level matching metric Knlg F1. Let E_{gen} be the entities mentioned in the generated summaries and E_{ref} be those in the reference summaries. We quantify the degree of overlap between

Instructor	DAE	QFE	R1	R2	RL
Initial Summary	0.699	1.837	76.03	66.34	74.11
FLAN T5 (Sup)	0.772	2.093	72.60	61.96	71.21
FLAN T5 (RL)	0.803	2.198	74.77	64.73	73.44
ChatGPT (10-shot)	0.834	2.583	*56.54*	*41.29*	*53.06*

Table 3: The ROUGE score and factual consistency scores of edited summaries with instructions generated by different instructors on DeFacto. We use ChatGPT (10-shot) as the editor model for all the results shown in the table.

Instructor	Knlg F1	R1	R2	RL
Initial Summary	44.15	40.28	16.65	33.23
FLAN T5 (Sup)	47.44	41.04	16.72	33.63
FLAN T5 (RL)	47.99	41.21	16.80	33.90
ChatGPT (5-shot)*	43.43	39.46	15.43	32.40
Oracle	60.80	43.08	18.37	35.24

Table 4: Knowledge coverage and ROUGE scores of edited summaries with instructions generated by different instructors on CNNDM. We use ChatGPT (zero-shot) as the generator model (to produce Initial Summary) and editor model. * We reduce the number of examples in the prompt if it exceeds the length limit (4k tokens).

Model	Knlg F1	R1	R2	RL
Initial Summary	44.15	40.28	16.65	33.23
Edit Iter 1	47.99	41.21	16.80	33.90
Edit Iter 2	48.65	41.18	16.69	33.88
Edit Iter 3	48.99	41.14	16.63	33.83
Edit Iter 1 (1&2)	48.08	41.25	16.91	33.94
Edit Iter 2 (1&2)	48.87	40.62	16.60	33.45
Edit Iter 3 (1&2)	49.20	41.15	16.87	33.86

Table 5: Iterative editing on CNNDM. The second block shows the results of the model fine-tuned on the data in the first iteration only, and the bottom block shows that of the model fine-tuned on the data in the first and second iterations.

the two by

$$\text{Knlg F1} = \frac{2\text{Knlg}_\text{p} \times \text{Knlg}_\text{r}}{\text{Knlg}_\text{p} + \text{Knlg}_\text{r}}, \text{ where}$$

$$\text{Knlg}_\text{p} = \frac{|E_\text{ref} \cap E_\text{gen}|}{|E_\text{gen}|}, \ \ \text{Knlg}_\text{r} = \frac{|E_\text{ref} \cap E_\text{gen}|}{|E_\text{ref}|}.$$

By maximizing this overlap, the instructor aims to produce summaries that effectively cover pertinent information as indicated by the reference.

Settings: We use the summaries generated by ChatGPT as the initial summaries.[6]. And we employ FlanT5-large (700M) as the instructor model for predicting keywords, using both the original document and the initial summaries generated by ChatGPT as input. Supervised training is performed using oracle keyword lists specifying which keywords to add and remove. Subsequently, the model undergoes editor-steered reinforcement learning fine-tuning, as detailed in Section 3, using a subset of 10,000 training examples from the dataset for efficiency.

Results: The results of our experiments, presented in Table 4, demonstrate the effectiveness of our instructor model in enhancing knowledge coverage, indicated by both entity matching and ROUGE scores. In a zero-shot setting, ChatGPT exhibits strong performance as a summarizer. Importantly, when provided with Oracle instructions, ChatGPT also demonstrates a robust capacity to correct and refine initial summaries in accordance with the specified instructions.

The integration of instructions generated by our trained instructor model leads to remarkable improvements in knowledge coverage, indicating that the summaries better align with user preference (comparing FLAN T5 (Sup) with Initial Summary). Moreover, we observe that the reinforcement learning fine-tuning process(FLAN T5 (RL))further improves the model's performance, resulting in moderate but meaningful gains in the evaluated metrics.

In contrast, when we explore a few-shot setting in which ChatGPT directly generates instructions without the use of the trained instructor(ChatGPT (5-shot)), the edited summaries exhibit a decline in performance. Specifically, both Knlg F1 and ROUGE scores are lower than those of the initial summaries, suggesting limitations in ChatGPT's ability to generate effective instructions for summary editing to better align with users' preference.

Overall, these findings underscore the value of

[6]The dataset is released, and can be found in the Github repo.

Initial Summary	A former corrections officer was punched by a young man on a plane after he asked him to stop using foul language. The former officer then took the young man down and held him until police arrived. Source: Daily Mail
Oracle Instruction	<Add> Chad Hurst <remove> Daily Mail
Human-written Reference	Chad Hurst of Salt Lake City, Utah was sucker punched by a plane passenger when they landed in the city Sunday . This after Hurst asked the young man to stop using foul language following their flight . Hurst, a former corrections officer, then took down the man and pinned his arms behind his back while waiting for law enforcement . The young man, who has still not been named by police, was charged with assault and public intoxication .
Predicted Instruction	<Add> Chad Hurst <remove> Daily Mail
ChatGPT-edited Summary	Chad Hurst, a former corrections officer from Salt Lake City, Utah, was punched by a young man on a plane after he asked him to stop using foul language. Hurst calmly took the young man down and held him until police arrived. The young man was charged with assault and public intoxication. Hurst's training as a former corrections officer taught him to never punch back but to control the situation and take the person down.

Table 6: An example from the CNNDM dataset.

our instructor as a powerful intermediary for guiding large language models such as ChatGPT in editing summaries to more closely adhere to user preference.

5 Discussion

5.1 Iterative Editing

In addition to performing one-step editing, we conducted experiments to explore the effectiveness of iterative editing on the CNNDM dataset[7]. The results of the iterative editing experiments are presented in Table 5. Utilizing reinforcement learning (RL) training based solely on data from the first iteration, we observed an improvement in the coverage of the edited summaries over the iterative editing process. We further fine-tuned the model using a mixture of data from both the first and second iterations, which leads to improved performance, as evidenced by enhanced knowledge F1 in the iteratively edited summaries.

5.2 Qualitative Examples

We show examples from the CNNDM dataset in Table 6. The instructor model can correctly detect the user's expectation and produce the editing instruction. ChatGPT is capable to edit the initial summary based on the given instruction, serving as an editor. [8]

[7]We did not conduct similar experiments on the DeFacto dataset because, for the majority of data examples, only one editing step is required to transition from the initial summary to the human-edited summary

[8]Examples from DeFacto are shown in the appendix.

6 Related Work

6.1 Text Editing

Post-editing techniques have been extensively studied in various NLP tasks, including sentence fusion (Malmi et al., 2019), style transfer (Reid and Zhong, 2021), and wiki-editing (Reid and Neubig, 2022; Faltings et al., 2021). These methods involve micro-defined operations such as insertion, deletion, and replacement. However, they often require a substantial amount of human-labeled data or complex editing chains. In contrast, our work focuses on abstract-level text editing using natural language instructions, leveraging the capabilities of large language models like ChatGPT. Similarly, Liu et al. (2022b) propose an approach involving a critic model for feedback generation and an editor model for revising initial summaries. We extend this approach by formalizing it as an iterative editing pipeline and enhancing it with inference-only language models and an editor-steered instructor.

Recently, (Liu et al., 2022a) introduced a novel training paradigm that aligns generated text with human values through a dynamic programming-derived chain-of-edits. However, this method requires additional fine-tuning of the language model, which may be impractical for models with limited resources and accessibility.

In another line of work, Welleck et al. (2022) proposed a framework that decomposes the original generation process into generator and corrector components, where the corrector is trained through online training to iteratively refine imperfect generations. Our work differs from them by decomposing the generation process into three components: the generator, the instructor, and the editor. This decomposition allows us to utilize large models for complex generation and correction tasks, while employing smaller models to predict user-specific editing instructions.

In parallel to our research, Madaan et al. (2023) propose a similar generation pipeline aimed at iteratively refining the generated output. However, their approach differs in that they utilize the same large language model (with varying prompts) for generating the initial output, providing feedback, and editing the output based on the received feedback, without considering any user-specific feedback. In contrast, our focus in this paper is on aligning the generated output more closely with user needs, guided by a trained instructor.

6.2 Large Language Models

The field of natural language processing has witnessed significant advancements in the realm of large language models (LLMs) (Chowdhery et al., 2022; Zhang et al., 2022; Thoppilan et al., 2022), leading to the creation of models that exhibit extraordinary language processing capabilities. Among these models, the GPT family (Brown et al., 2020) stands as a prominent example, earning widespread recognition for its versatile performance across different language-related tasks.

The introduction of instruction tuning (Wei et al., 2021) has further catalyzed the enhancement of language models, particularly when trained with human instructions (Sanh et al., 2021). Notably, this approach has resulted in substantial improvements, especially within the context of zero-shot and few-shot learning. InstructGPT (Ouyang et al., 2022), which employs the Reinforcement Learning from Human Feedback (RLHF) training paradigm, exemplifies this trend, enabling models to effectively follow human instructions and providing a foundational basis for our current work.

The recent release of LLAMA (Touvron et al., 2023) has further expanded opportunities for exploration in this area, as researchers have begun to train or fine-tune models using task-augmented datasets by GPT models (Wang et al., 2022).

Distinct from the aforementioned research efforts, our work introduces the tri-agent pipeline, a novel paradigm that capitalizes on the capabilities of large language models for downstream tasks. Uniquely, our approach is designed to optimize performance while minimizing computational resource demands and accommodating limited access to large language models (e.g., API-only access).

6.3 Summarization with LLM

Before the advent of LLMs, a prevalent approach to the text summarization task involved pre-training models on a substantial corpus using task-focused objectives, followed by fine-tuning on task-specific datasets. This paradigm demonstrated effectiveness in text summarization and was adopted by models such as PEGASUS (Zhang et al., 2020), Primera (Xiao et al., 2021), and Z-Code++ (He et al., 2022). However, recent studies (Goyal et al., 2022; Zhang et al., 2023) have revealed that the application of GPT-3 (Brown et al., 2020) and InstructGPT (Ouyang et al., 2022) to news summarization tasks in zero-shot settings yields results that are not only preferred by human evaluators over those of supervised models, but are also more favorable than the reference summaries themselves.

These findings suggest a direction for the text summarization task. Rather than training supervised summarizers on potentially suboptimal reference summaries, it may be more efficient to leverage LLMs, and focus on editing their outputs to align with user requirements, which is also in-line with the tri-agent pipeline proposed in this work.

7 Conclusion and Future Work

In this paper, we introduce a novel generation paradigm that decomposes the generation process into three distinct components: the generator, the instructor, and the editor. Our approach is specifically designed to harness the capabilities of large language models, while accounting for constraints such as limited access and computational resources, and to facilitate the customization of generated content to align with user preference. Through a series of pilot experiments on the task of text summarization, we find that large language models, exemplified by ChatGPT, can effectively serve as editors, achieving performance levels comparable to supervised editing models when provided with human-written instructions. Nevertheless, it is still challenging for the large language models to generate instructions that are well-aligned with human-authored instructions.

To address this challenge, we employ a smaller model as the instructor, which is trained with editor-steered reinforcement learning (RL) with rewards based on the quality of the edited summaries. Our experimental results demonstrate the efficacy of this approach in guiding the editor (ChatGPT) to produce summaries that are more closely aligned with user expectations.

Looking ahead, future work will involve extending our experiments to other tasks, such as wiki-editing (Reid and Neubig, 2022), news-editing (Spangher et al., 2022), and mathematical problem synthesis (Welleck et al., 2022). Additionally, we may generate more instruction data using the self-instruct technique (Wang et al., 2022) to train a better instructor.

Limitations

While our proposed generation pipeline aims to improve the alignment of large language model outputs with user preference, we acknowledge the limitation of resource constraints in our study. As a result, we focus our experiments solely on ChatGPT, which has demonstrated top performance across a range of tasks. However, future work should explore its applicability and performance with other large language models as well. Furthermore, it is important to note that, like all large language models, our system's output may still exhibit issues such as hallucination and bias. While our pipeline partially addresses these concerns, we cannot guarantee that the results are completely free from hallucination and bias.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, Seattle, United States. Association for Computational Linguistics.

Felix Faltings, Michel Galley, Gerold Hintz, Chris Brockett, Chris Quirk, Jianfeng Gao, and Bill Dolan. 2021. Text editing by command. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5259–5274, Online. Association for Computational Linguistics.

Tanya Goyal and Greg Durrett. 2021. Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Pengcheng He, Baolin Peng, Liyang Lu, Song Wang, Jie Mei, Yang Liu, Ruochen Xu, Hany Hassan Awadalla, Yu Shi, Chenguang Zhu, Wayne Xiong, Michael Zeng, Jianfeng Gao, and Xuedong Huang. 2022. Z-code++: A pre-trained language model optimized for abstractive summarization.

Ruibo Liu, Chenyan Jia, Ge Zhang, Ziyu Zhuang, Tony Liu, and Soroush Vosoughi. 2022a. Second thoughts are best: Learning to re-align with human values from text edits. In *Advances in Neural Information Processing Systems*, volume 35, pages 181–196. Curran Associates, Inc.

Yixin Liu, Budhaditya Deb, Milagro Teruel, Aaron Halfaker, Dragomir Radev, and Ahmed H. Awadallah. 2022b. On improving summarization factual consistency from natural language feedback.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2023. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization.

Machel Reid and Graham Neubig. 2022. Learning to model editing processes. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3822–3832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Machel Reid and Victor Zhong. 2021. LEWIS: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944, Online. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207.

Alexander Spangher, Xiang Ren, Jonathan May, and Nanyun Peng. 2022. Newsedits: A news article revision dataset and a document-level reasoning challenge.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. 2022. Lamda: Language models for dialog applications. *CoRR*, abs/2201.08239.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2021. PRIMER: pyramid-based masked sentence pre-training for multi-document summarization. *CoRR*, abs/2110.08499.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2023. Benchmarking large language models for news summarization.

A Prompts

We show the prompts used for summary editing and instruction generation in Table 7 and Table 8, respectively.

CNNDM
Summary: [initial summary] Document: [article] Rewrite the summary for the document, [instruction] New summary:
DeFacto
Document: [article] Summary: [initial summary] Instructions: [instruction] Edit the summary only following the instructions and only output the corrected summary. New summary:

Table 7: Prompts used for summary editing.

CNNDM
few-shot prompts $\times N$*, up to the length limit* Document: [article]$_i$ Summary: [initial summary]$_i$ Instructions: [instruction]$_i$ Document: [article] Summary: [initial summary] The summary may not cover the salient content, generate instructions to make the summary focus on salient content. The instructions should be chosen from the following formats: Delete content related to __. Add content related to __. No operation is needed. Only output the instructions without the corrected summaries, and make the instruction conservatively. Instructions:
DeFacto
few-shot prompts $\times 10$ Document: [article]$_i$ Summary: [initial summary]$_i$ The summary may contain some factual errors, generate the instructions to correct the summary. Instructions: Document: [article] Summary: [initial summary] The summary may contain some factual errors, generate the instructions to correct the summary. The instructions should be chosen from the following formats: Remove the information about __ from the summary. Add the information about __ to the summary. Replace the information about __ with the in-formation about __. Modify the information about __ in the summary. Rewrite the summary entirely by __. Only output the instructions without the corrected summaries, and make the instruction conservatively. Instructions:

Table 8: Prompts used for instruction generation

B Qualitative Examples

We show examples from the DeFacto dataset in Table 6. The instructor model can correctly detect the user's expectation and produce the editing instruction. ChatGPT is capable to edit the initial summary based on the given instruction, serving as an editor.

Initial Summary	The controversial Kudankalum nuclear power station in India's Tamil Nadu state has started generating electricity.
Human-written Instruction	Remove the information about the location of India's Tamil Nadu state from the summary.
Human-edited Summary	The controversial Kudankalum nuclear power station has started generating electricity.
Predicted Instruction	Remove the information about Tamil Nadu from the summary.
ChatGPT-edited Summary	The controversial Kudankalum nuclear power station has started generating electricity.
Initial Summary	Gunfire has been heard in Ivory Coast's second city of Bouaké, a day after soldiers mutinied over pay
Human-written Instruction	Remove the information about second from the summary.
Human-edited Summary	Gunfire has been heard in Ivory Coast city of Bouaké, a day after soldiers mutinied over pay.
Predicted Instruction	Remove the information about second from the summary.
ChatGPT-edited Summary	Gunfire has been heard in Ivory Coast's city of Bouak, a day after soldiers mutinied over pay.

Table 9: Examples from the DeFacto dataset.

C Software and Licenses

Our code is licensed under Apache License 2.0.
Our framework dependencies are:

- HuggingFace Datasets[9], Apache 2.0
- NLTK [10], Apache 2.0
- Numpy[11], BSD 3-Clause "New" or "Revised"
- Transformers[12], Apache 2.0
- Pytorch[13], Misc
- ROUGE [14], Apache 2.0
- Flan T5 [15], Apache 2.0
- ChatGPT [16], Proprietary

[9]https://github.com/huggingface/datasets/blob/master/LICENSE
[10]https://github.com/nltk/nltk
[11]https://github.com/numpy/numpy/blob/main/LICENSE.txt
[12]https://github.com/huggingface/transformers/blob/master/LICENSE
[13]https://github.com/pytorch/pytorch/blob/master/LICENSE
[14]https://github.com/google-research/google-research/tree/master/rouge
[15]https://huggingface.co/google/flan-t5-large
[16]https://openai.com/chatgpt