Constraint Text Revision Agent Via Iterative Planning and Searching

Hannan Cao and Hwee Tou Ng

National University of Singapore

Outline

- Motivation
- Observations
- Methods
- Dataset Construction
- Experimental Results
- Conclusion

Motivation

- Existing text revision system:
  - Provides writing suggestions based on user instruction, focusing on:
    - Single-sentence revision.
    - Unconstrained revision.

Motivation

- However, in real-world application:
  - Users expect a text revision system that:
    - Revises text at the paragraph level.
    - Adheres to specific constraints (e.g., sentence structure, word limits, length restrictions).
  - We name this task Constrained Text Revision (CTR).

Motivation

- Furthermore, CTR has diverse applications.
  - ➢ Plain text revision, LaTeX document revision.
  - ➢ Therefore, designing a universal CTR system for all use cases is challenging.
- Aim to design a text revision agent:
  - Develop an intelligent agent capable of performing **paragraph-level** text revision by following **various constrained instructions**. The agent should be **adaptable to diverse use cases** with ease.

Observations

- LLM's CTR ability (both text quality and constraint adherence) benefits from:
  - **Structured planning**
  - LLMs benefit more from human's revision plan

	PPL↓	SOME↑	BART.↑
w/o Plan	34.58	88.91	-2.46
w/ GPT-4o Plan	23.64	91.67	-1.92
w/ Human Plan	**21.31**	**93.28**	**-1.49**

Table 1: Revised text quality under three conditions: without plans (**w/o Plan**), with GPT-4o-generated plans (**w/ GPT-4o Plan**), and with human-labeled plans (**w/ Human Plan**). SOME is reported in %, and BART. represents the BARTScore.

	L1	L2	L3	L4
w/o Plan	68.00	61.00	53.66	46.50
w/ GPT-4o Plan	71.00	67.00	61.00	54.00
Gain	**+3.00**	**+6.00**	**+7.34**	**+7.50**

Table 2: Constraint adherence accuracy (%) under different constraints for two settings: without plans (**w/o Plan**) and with GPT-4o-generated plans (**w/ GPT-4o Plan**). **Gain**: the performance gain with the plan.

Observations

- LLM's CTR ability (both text quality and constraint adherence) benefits from **iterative revisions**.

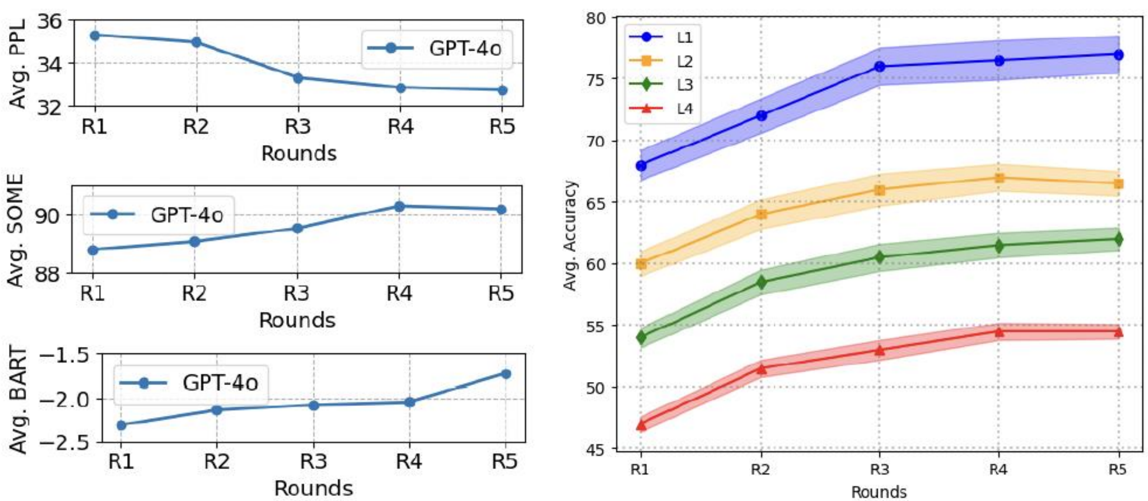
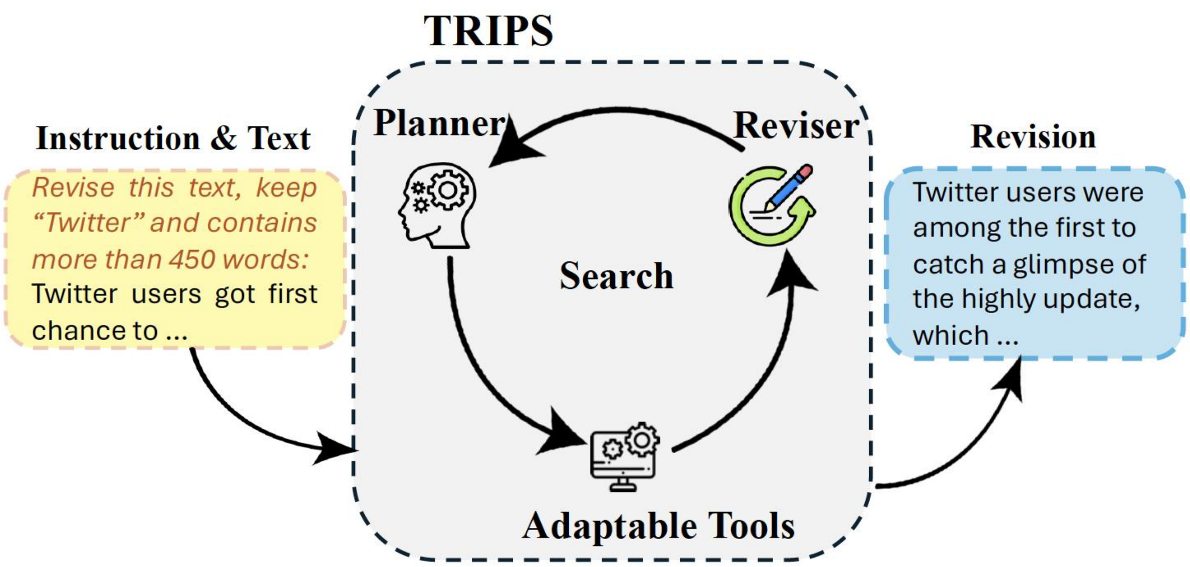


Figure 3: **Left:** Average PPL, SOME, and BARTScore for revised text across five revision rounds (R1–R5). **Right:** Average accuracy for different revision rounds.

Method

- Design **TRIPS**, a constraint **T**ext **R**evision agent via **I**terative **P**lanning and **S**earching for CTR:



Method

- TRIPS operate iteratively in two phases:
  - Planning:
    - Utilizes a planner to formulate tool usage and revision strategies tailored to different scenarios.
  - Searching:
    - Employs selected tools to guide the search algorithm in identifying optimal revision plans for the reviser (i.e., a vanilla LLM).

Method

- Planner
  - ➢ Requires understanding the constraints to formulate:
    - ➢ Tool usage
    - ➢ Text revision plans
  - ➢ However, constrained revisions often involve numerical symbols (Jiang et al., 2024), which LLMs frequently misinterpret (Chen et al., 2024).

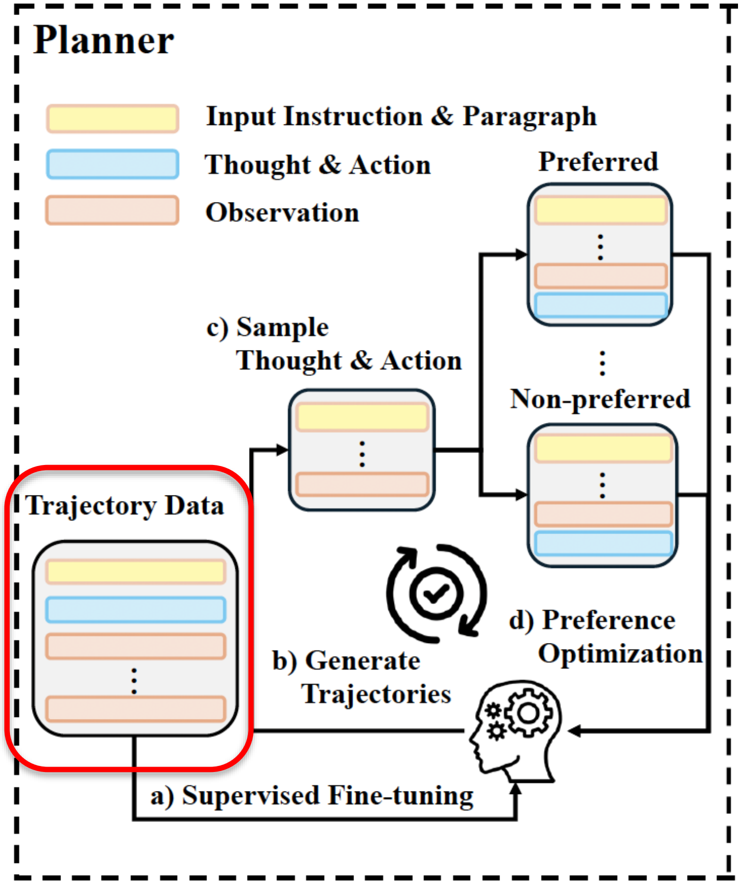
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. *In ACL 2024.*
- Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. Benchmarking large language models on controllable generation under diversified instructions. *In AAAI 2024.*

Method - Planner

- Build the planner in two steps:
  - Generate Synthetic Trajectories with GPT-4o through in-context learning (ICL)
  - Use the trajectory to fine-tune LLMs through:
    - Supervised Fine-Tuning (SFT)
    - Iterative self-training alignment

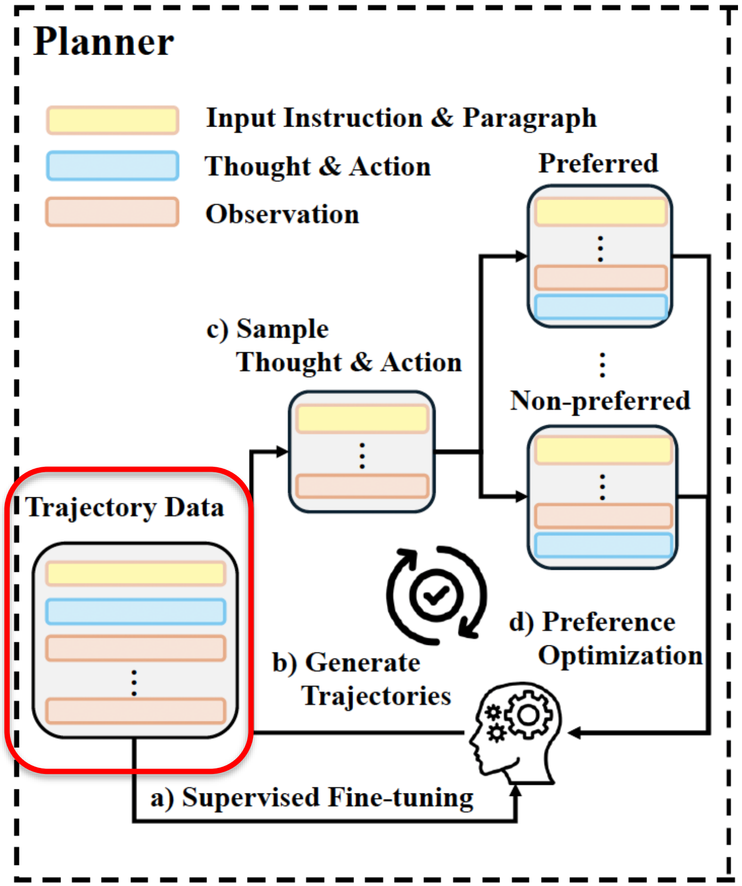
Method - Planner

- Synthetic Trajectory Generation
  - ➢ Leverage GPT-4o to generate tool usage and text revision planning trajectories via ICL.
    - ➢ Use human-labeled **revision plans** as in-context examples.
    - ➢ Adapt the ReAct format.



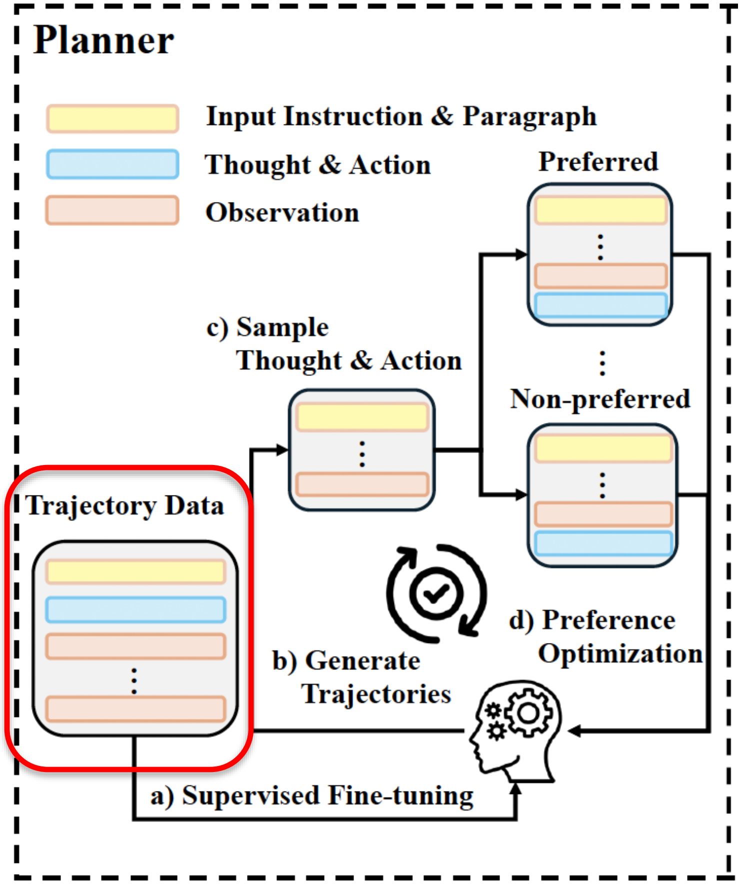
Method - Planner

- Synthetic Trajectory Generation
  - ReAct format
    - ➢ ***Observation***: Input text and instruction.
    - ➢ ***Thought***: Identify constraints and areas for improvement.
    - ➢ ***Action***: Form tool usage and text revision plans.
    - ➢ Revised text and feedback form the ***new observation***.



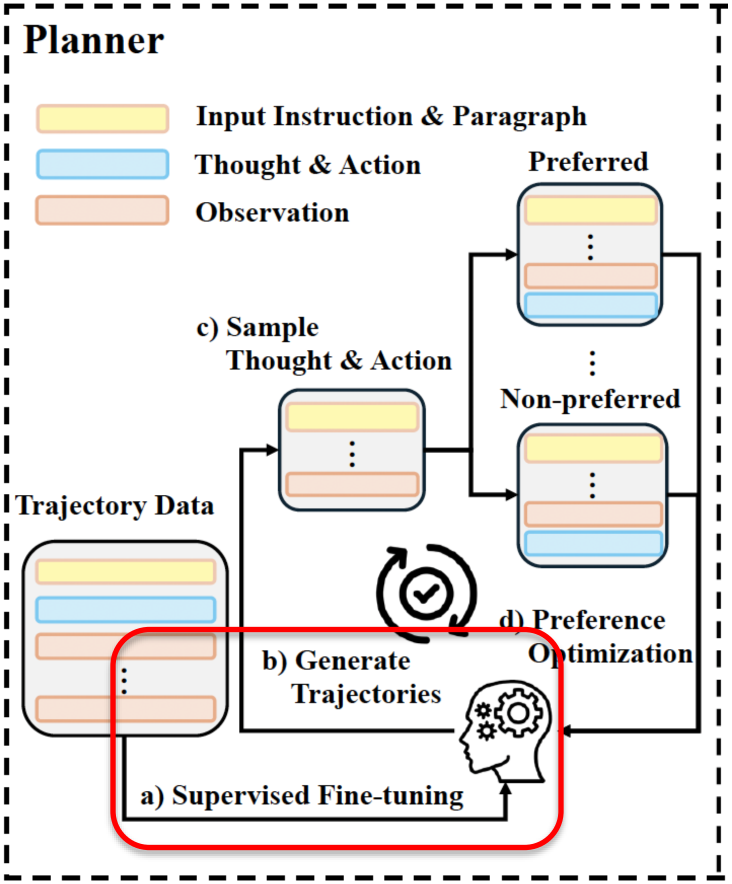
Method - Planner

- Synthetic Trajectory Generation
  - Iterate the above steps until:
    - Reaching the maximum number of iterations or
    - Until further iterations no longer improve the revision quality.



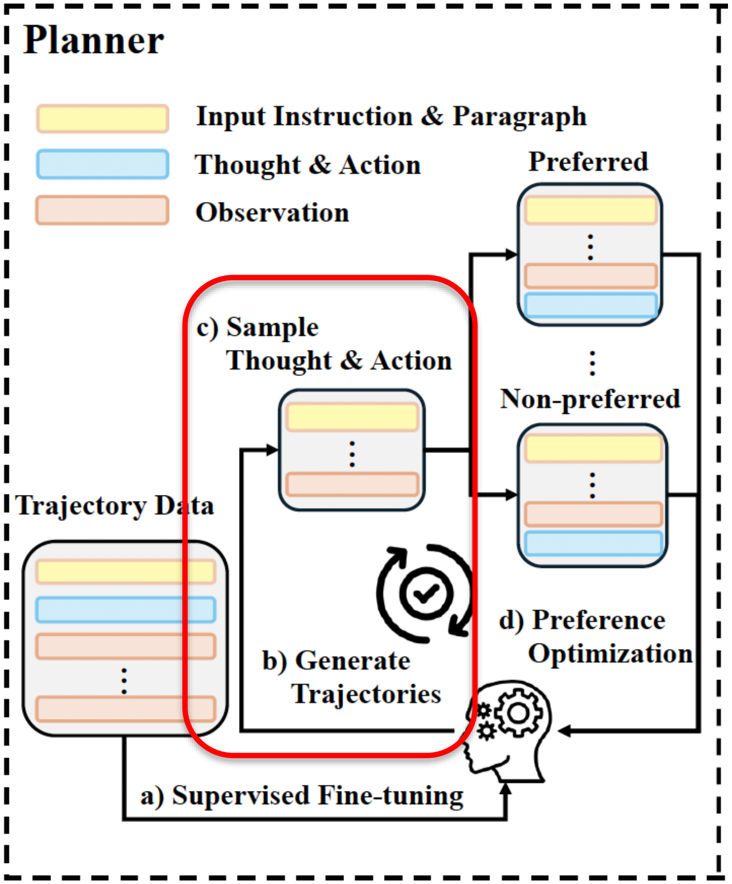
Method - Planner

- Use synthetic trajectory to build an initial planner via SFT.
- Create new trajectory H_i with the initial planner by generating steps up to i.
- Sample multiple thought and action pairs based on H_i.
- Evaluate the action with a scoring function to create the preference data.
- Using this preference data to further optimize the planner.



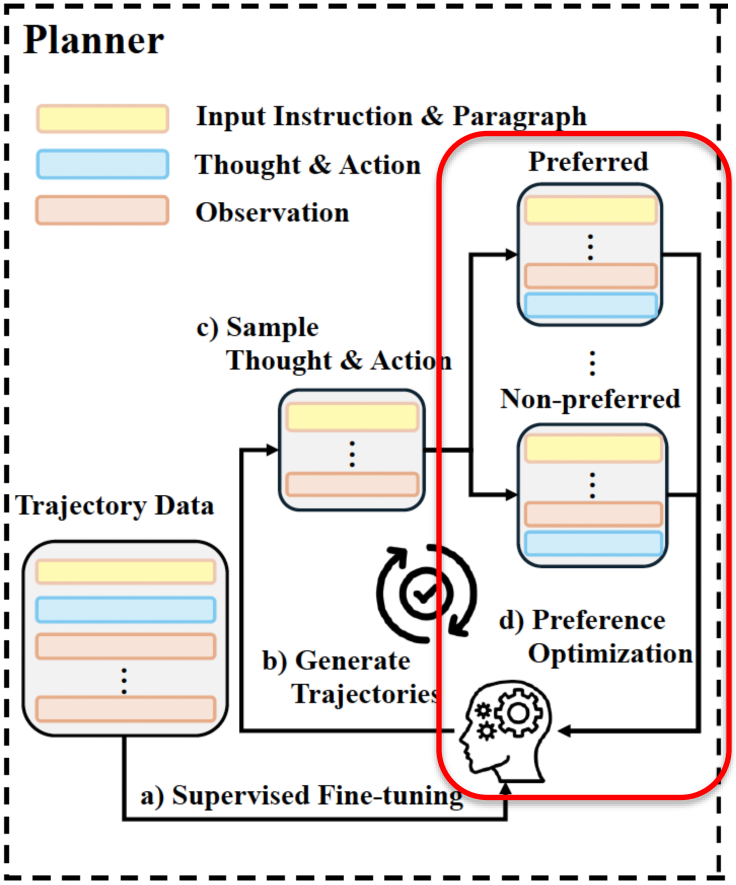
Method - Planner

- Use synthetic trajectory to build an initial planner via SFT.
- Create new trajectory H_i with the initial planner by generating steps up to i.
- Sample multiple thought and action pairs based on H_i.
- Evaluate the action with a scoring function to create the preference data.
- Using this preference data to further optimize the planner.



Method - Planner

- Use synthetic trajectory to build an initial planner via SFT.
- Create new trajectory H_i with the initial planner by generating steps up to i.
- Sample multiple thought and action pairs based on H_i.
- Evaluate the action with a scoring function to create the preference data.
- Using this preference data to further optimize the planner.



Method - Planner

- Action (a_{i+1}) scoring function:

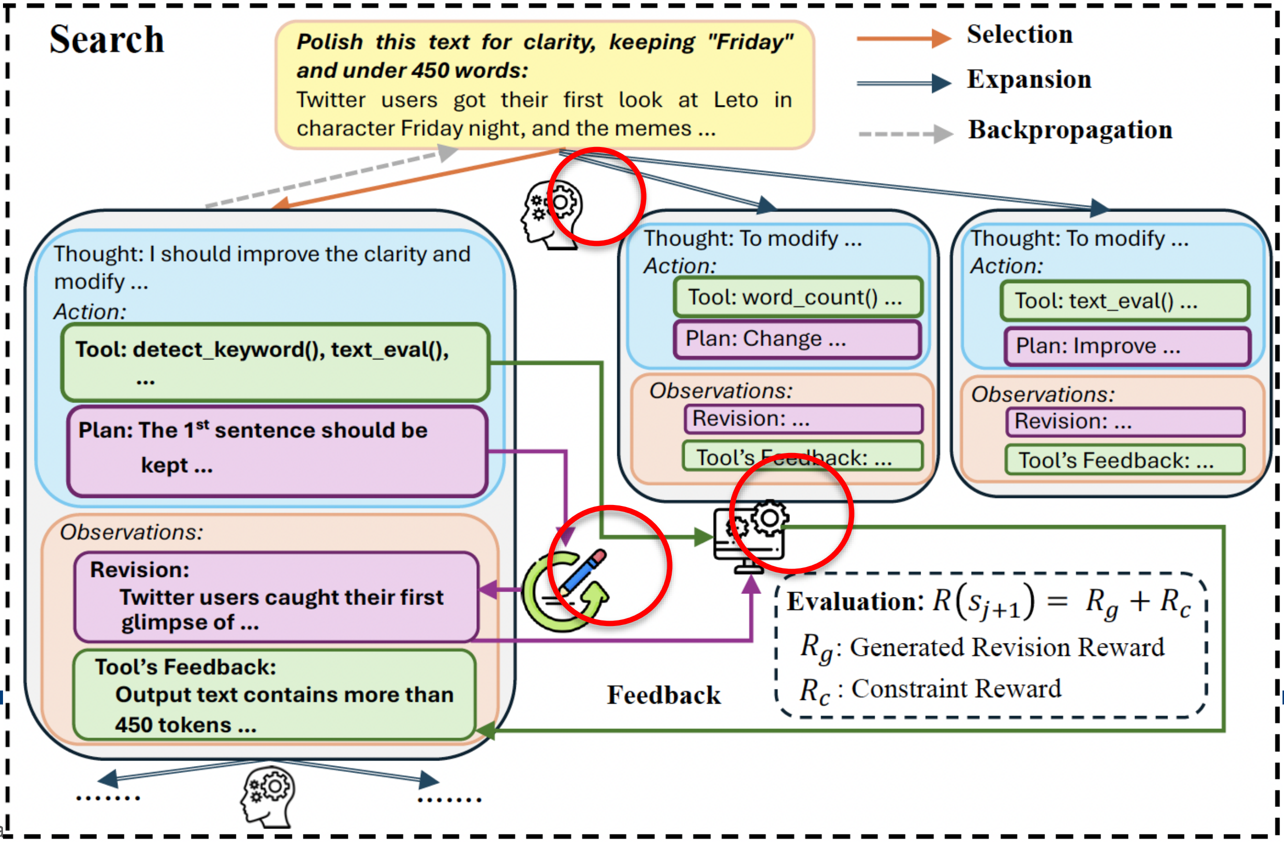
$$S_a(a_{i+1}) = \lambda_v \cdot S_v + \lambda_r \cdot S_r + \lambda_c \cdot S_c,$$

  - S_v: Tool usage quality, S_r: Revision quality; S_c: Constraint adherence quality.
  - λ_v, λ_r, and λ_c: respective weight.
- Preference Optimization:
  - Highest scoring action with its thought form the winning response w_{i+1}.
  - Use L_P, containing both SimPO (Meng et al., 2024) and cross entropy computed on the winning response to optimize the planner:

$$\begin{aligned}\mathcal{L}_P =\ & \mathcal{L}_{SimPO} - \log \pi_n(w_{i+1}|\mathcal{H}_i) \\ =& -\log \sigma \left(\frac{\beta \log \pi_n(w_{i+1}|\mathcal{H}_i)}{|w_{i+1}|} - \frac{\beta \log \pi_n(l_{i+1}|\mathcal{H}_i)}{|l_{i+1}|} - \gamma \right) \\ & - \log \pi_n(w_{i+1}|\mathcal{H}_i),\end{aligned}$$

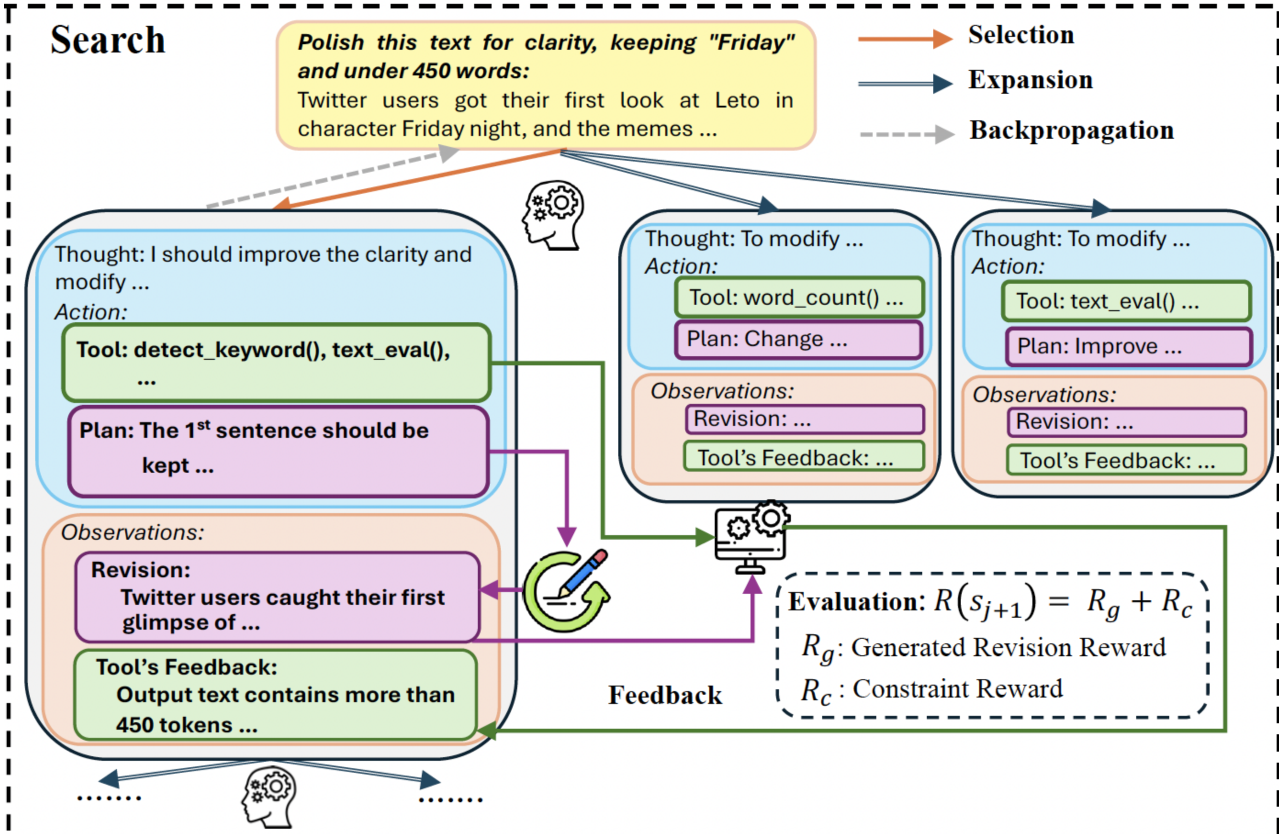
Method – Search

- Propose a ***Tool-Guided Monte Carlo Tree Search*** (TG-MCTS): A novel approach that seamlessly integrates a **planner**, **reviser**, and **adaptable tools**, enabling efficient adaptation to diverse CTR scenarios.



Method – Search

- TG-MCTS extends traditional MCTS with two key components:
  - ➢ Tool-Guided Expansion
  - ➢ Tool-Based Evaluation



Method – Search

- TG-MCTS:
  - Each j-th node in the tree is defined as:

$$s_{\mathrm{j}} = \{o_j, H_j, N(s_j), V(s_j)\}$$

- o_j: Observation at j-th node, containing the revised text y_j and feedback.
- H_j: Historical trajectory to the current node.
- $N(s_j)$: Node's visit count.
- $V(s_j)$: Node's value score, corresponds to the expected reward of s_j.

Method – Search

- TG-MCTS iteratively performs: a) Selection; b) Tool-Guided Expansion; c) Tool-Based Evaluation; d) Backpropagation

  - Selection: TG-MCTS selects a node based on the Upper Confidence Bounds applied to Trees (UTC) score:

$$UCT(s_j) = V(s_j) + \alpha \sqrt{\frac{\ln N(p)}{N(s_j)}}, \qquad (3)$$

p: parent node of s_j, α hyper-parameter, balancing between exploitation ($V(s_j)$) and exploration ($N(s_j)$)

Method – Search

- Tool-Guided Expansion:
  - **Revise**:
    - Expand the selected node by generate a set of actions a_{j+1}.
    - Generate new revision y_{j+1} based on the revision plan with the reviser (π_θ): $y_{j+1} = \pi_\theta(a_{j+1}, y_j)$
  - **Feedback**:
    - Use the selected tools to provide feedback for y_{j+1}, containing:
      - Revision feedback - suggestions for improving the revision.
      - Constraint feedback - suggestions for improving the constraint adherence.

Method – Search

- Tool-Based Evaluation:
  - Compute the expected reward $R(s_{j+1})$ for the new node s_{j+1} using the selected tools, $R(s_{j+1}) = R_g + R_c$:
    - R_g: Generated revision reward
    - R_c: Constraint reward
- Backpropagation:
  - Updates the values and visit counts of all nodes along the path from the root node to its parent nodes s_k $(0 \leq k \leq j)$

$$N_{\text{new}}(s_k) = N_{\text{old}}(s_k) + 1, \tag{4}$$

$$V_{\text{new}}(s_k) = \frac{V_{\text{old}}(s_k) N_{\text{old}}(s_k) + R(s_{j+1})}{N_{\text{new}}(s_k)}, \tag{5}$$

Dataset Construction

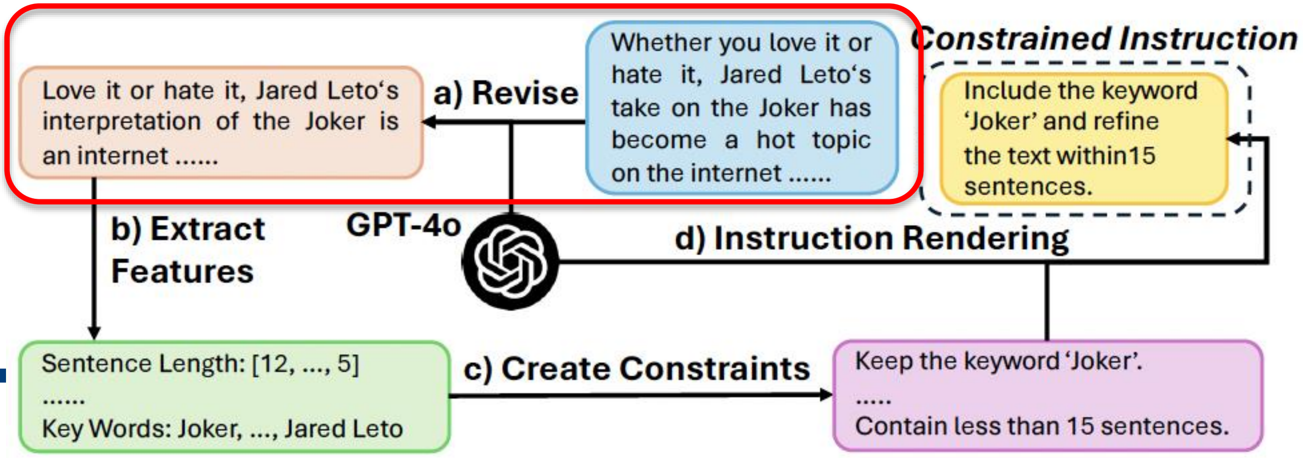
- We introduce ***ConsTRev*** for constrained text revision task, with a focus on:
  - Paragraph-level revision
  - Multiple-level, complex, verifiable, and valid text revision constraints.
    - Contains L0 domain: text paired with text revision instructions without constraints.
    - Contains L1 – L4 domain: each containing text paired constrained text revision instructions containing one to four constraints, respectively.

Dataset Construction

- Data Source:
  - A curated selection of **500 texts** from diverse sources:
    - Academic papers
    - WikiHow articles
    - Human-written stories
  - Each text contains 350 to 1000 words.
  - Five domains (L0–L4), each containing 100 texts.

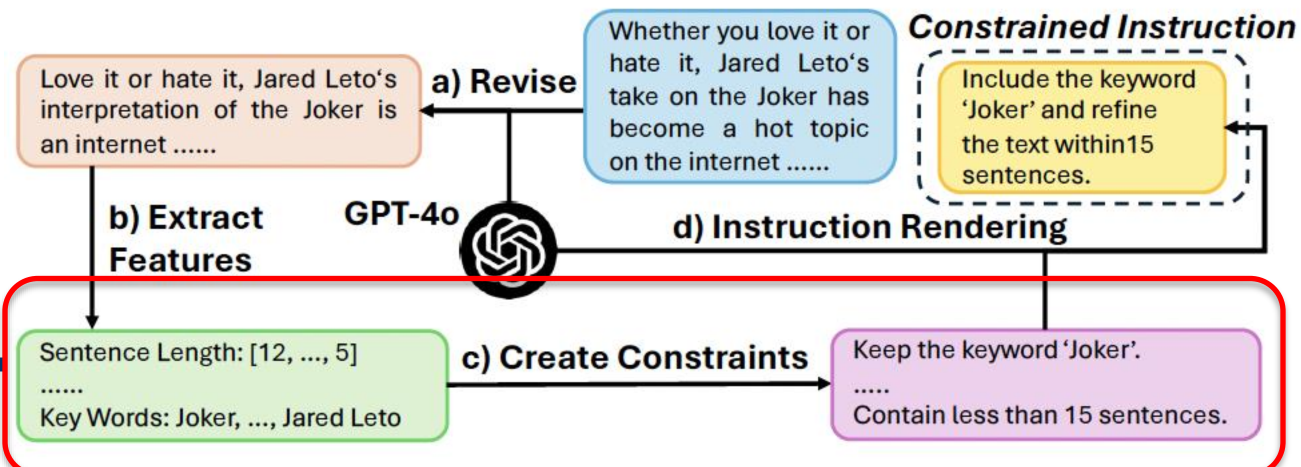
Dataset Construction

- Constrained Instruction Creation
  - Use GPT-4o to revise the selected text.
  - Extract relevant features and structure constrained instructions via program template.
  - Combine multiple (0-4) constrained instructions into a set.
  - Use GPT-4o to refine and improve fluency for more natural and effective instructions.



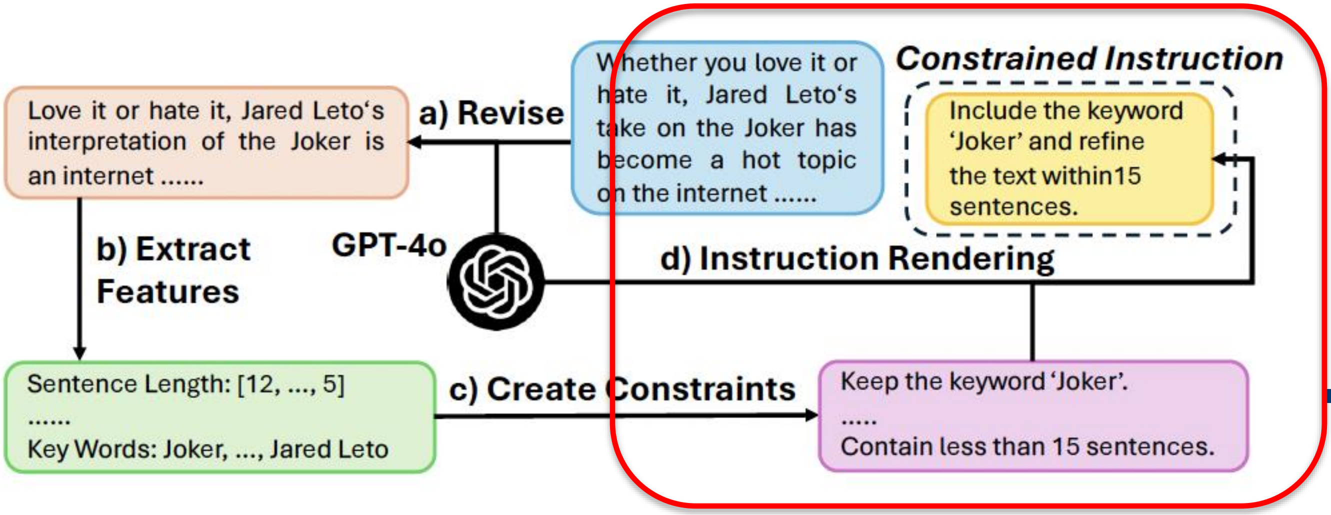
Dataset Construction

- Constrained Instruction Creation
  - Use GPT-4o to revise the selected text.
  - Extract relevant features and structure constrained instructions via program template.
  - Combine multiple (0-4) constrained instructions into a set.
  - Use GPT-4o to refine and improve fluency for more natural and effective instructions.



Dataset Construction

- Constrained Instruction Creation
  - Use GPT-4o to revise the selected text.
  - Extract relevant features and structure constrained instructions via program template.
  - Combine multiple (0-4) constrained instructions into a set.
  - Use GPT-4o to refine and improve fluency for more natural and effective instructions.



Experiment – Dataset & Model

- Dataset
  - We evaluate **TRIPS** on **ConsTRev** across 5 domains (L0- L4)

- Model:
  - We develop two systems:
    - **TRIPS-3.1**:
      - Use Llama-3.1-8B-Instruct as the reviser
    - **TRIPS-4o**:
      - Use GPT-4o as the reviser
    - Both systems use Llama-3.1-8B-Instruct as the base model for constructing the planner.

Experiment – Baseline & Results

- Compare against **SOTA** text revision systems (CoEDIT-C) and CTG (Evol-Ins & Conifer)
- GPT-4o/LLama3.1 baselines:
  - **Direct** Prompting, **CoT**, Human-Plan (**Plan**), Iterative Revision (**Iter**)
- Results: TRIPS-3.1/4o reaches the **best text quality among baselines**.

System		L0 PPL↓	L0 SOME↑	L0 BART.↑
CoEDIT-C		38.82	87.32	-2.16
LLaMA 3.1	Direct	29.69	83.61	-4.97
	CoT	27.38	84.58	-4.77
	Plan	27.31	84.18	-4.58
	Iter	<u>26.55</u>	84.21	-4.52
	TRIPS-3.1	**25.82**	**88.96**	-1.92
GPT-4o	Direct	35.92	87.61	-2.18
	CoT	36.16	88.62	-2.21
	Plan	35.24	88.14	<u>-1.87</u>
	Iter	34.74	88.21	-1.89
	TRIPS-4o	33.07	<u>88.80</u>	**-1.76**

Table 4: Performance on the ConsTRev L0 domain. SOME is shown in %. BART. denotes the BARTScore. The best and second-best results are highlighted in **bold** and <u>underline</u>, respectively.

Experiment - Results

➢ TRIPS-3.1/4o achieves the best performance in **constrained instruction following**.

System	L1				L2				L3				L4			
	Cons.	Text Quality			Cons.	Text Quality			Cons.	Text Quality			Cons.	Text Quality		
	Acc.↑	PPL↓	SOME↑	BART.↑	Acc.↑	PPL↓	SOME↑	BART.↑	Acc.↑	PPL↓	SOME↑	BART.↑	Acc.↑	PPL↓	SOME↑	BART.↑
Evol-Ins	57.00	32.79	86.87	-2.32	53.0	39.12	87.83	-2.23	51.33	38.29	87.79	**-1.17**	42.00	31.54	87.24	-1.94
Conifer	51.00	39.16	85.71	-3.42	59.0	46.04	87.79	-2.88	52.00	43.74	88.28	-2.48	44.25	41.11	88.42	-2.65
LLaMA 3.1 8B Instruct																
Direct	58.00	30.92	83.34	-4.46	59.5	33.95	87.41	-3.74	50.33	34.20	88.31	-2.54	42.25	31.38	91.13	-2.55
CoT	60.00	30.15	84.23	-5.19	57.5	34.72	87.85	-4.68	51.00	32.84	88.41	-3.81	46.00	30.73	**91.87**	-3.81
Plan	62.00	29.56	85.14	-4.08	61.5	30.21	87.85	-3.38	54.66	29.33	88.61	-2.34	46.25	28.98	91.41	-3.22
Iter	65.00	29.23	83.74	-3.82	63.5	29.96	88.22	-3.32	57.33	28.22	88.82	-3.18	48.25	28.37	91.16	-3.18
TRIPS-3.1	83.00	**27.49**	**89.00**	-1.95	80.0	**29.80**	88.74	**-1.86**	80.00	**28.18**	**89.00**	-2.00	72.75	**27.82**	88.44	-1.80
GPT-4o																
Direct	69.00	51.91	86.41	-2.23	61.5	53.37	87.56	-1.95	54.33	50.61	89.00	-1.98	47.00	46.87	88.64	-1.93
CoT	68.00	50.55	86.21	-2.06	63.0	49.71	88.10	-1.93	55.66	48.83	87.89	-1.92	48.75	45.43	88.78	-1.92
Plan	72.00	42.05	86.75	-2.01	66.5	44.68	88.06	-1.91	60.00	42.89	88.07	-1.98	53.75	43.41	88.61	-1.92
Iter	77.00	40.78	86.95	-2.41	67.5	43.84	88.32	-1.92	62.33	42.28	87.12	-1.93	54.75	44.64	88.73	-1.84
TRIPS-4o	**85.00**	32.52	87.11	**-1.82**	**83.0**	39.11	**88.84**	-1.87	**82.66**	34.45	88.63	-1.87	**76.50**	32.87	88.82	**-1.72**

Table 3: Performance on ConsTRev across L1-L4 domains. **Cons.** denotes constraint adherence quality, **Acc.** denotes accuracy, and **BART.** denotes the BARTScore. Both Acc. and SOME are shown in %. The best results are **bolded**, and the second-best results are underlined across all domains.

Analysis

- TRIPS-4o vs GPT-4o(Iter)(i.e., the best performing baseline) under LLM-as-a-Judge evaluation:
  - Evaluate 100 outputs fromTRIPS-4o and GPT-4o(Iter)
  - Results indicate that **TRIPS-4o consistently outperforms GPT-4o(Iter)**

	TRIPS-4o	GPT-4o
F (↑)	**4.93**	4.87
C (↑)	**4.82**	4.67
G (↓)	**0.02**	0.06

	# Cases
F	67
C	72
G	85

Table 5: LLM-as-a-Judge using GPT-4. **Left**: Average scores assigned by GPT-4. **Right**: Number of cases (**# Cases**) where TRIPS-4o outperformes GPT-4o.

Analysis

- Each components plays an important role in improving TRIPS' performance

System	L0		
	PPL↓	**SOME↑**	**BART.↑**
TRIPS-4o	**33.07**	**88.80**	-1.76
w/o Plan	34.93	88.16	-1.91
w/o Feedback	34.21	88.24	-1.88
w/o R_g	33.95	88.56	-1.82
w/o R_c	33.09	88.78	**-1.74**

Table 6: Revision quality on the ConsTRev L0 domain.

	L1	L2	L3	L4
TRIPS-4o	**85.00**	**83.00**	**82.66**	**76.50**
w/o Plan	76.00	65.50	60.66	54.25
w/o Feedback	79.00	69.00	62.00	56.00
w/o R_g	84.00	82.50	81.66	75.25
w/o R_c	81.00	73.00	68.33	62.75

Table 7: Constraint adherence accuracy on ConsTRev across L1 to L4 domains.

Analysis

- Preserving named entities during revision ensures the original meaning remains intact.
- TRIPS-4o achieves a higher named entity preservation rate compared to GPT-4o (Iter).

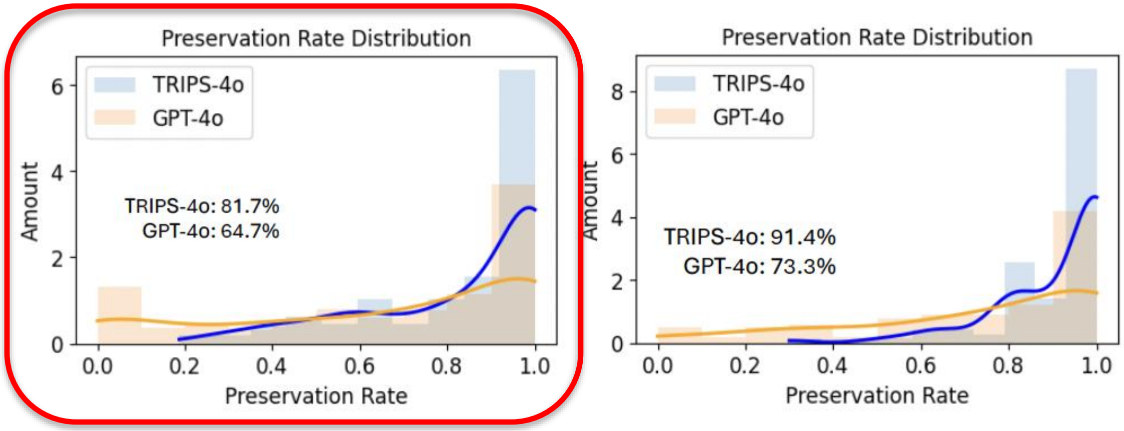


Figure 5: The preservation rate distribution. **Left:** Named entity. **Right:** LaTeX keyword.

Analysis

- TRIPS-4o can be easily extended to other use cases, like LaTeX revision
  - Producing revisions:
    - Containing fewer error
    - Better text quality

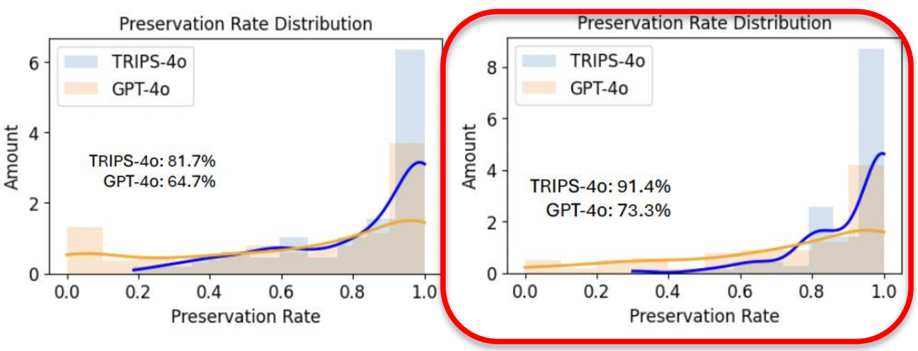


Figure 5: The preservation rate distribution. **Left:** Named entity. **Right:** LaTeX keyword.

	AvgCE. ↓	Text Quality		
		PPL ↓	SOME ↑	BART ↑
GPT-4o	0.24	48.72	85.37	-1.92
TRIPS-4o	**0.06**	**35.65**	**88.21**	**-1.61**

Table 8: Revised text generated by TRIPS-4o and GPT-4o. **AvgCE.**: the average compilation error. **Text Quality**: the quality of the revision after compilation.

Analysis

- Our planner largely surpass GPT-4o and its base model
- Our self-training alignment method effectively enhances the planner's tool usage performance across iterations.

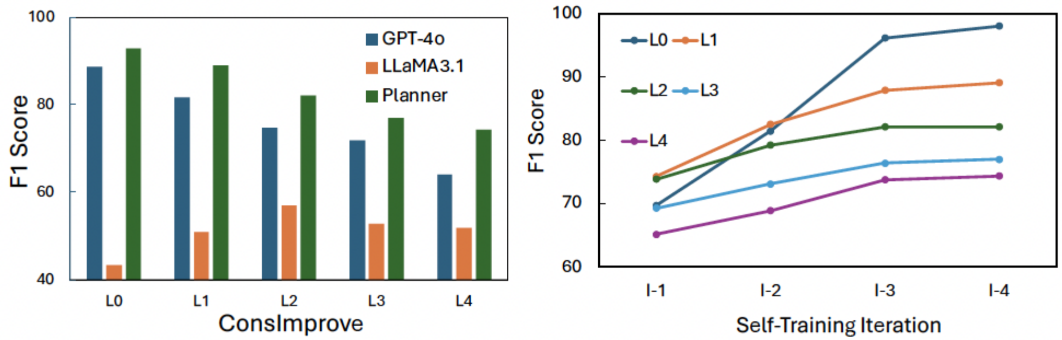


Figure 6: F_1 score (in %) for tool usage quality. **Left:** Tools usage generated by GPT-4o, Llama-3.1-8B-Instruct, and the planner. **Right:** Tool usage quality across four iterations (I-1 to I-4).

Conclusion

- We introduce **Constrained Text Revision (CTR)**, a novel task, along with **ConsTRev**, a dedicated dataset.
- We formulate **CTR** as an **iterative planning and searching problem** and propose **TRIPS** as a solution.
- TRIPS significantly outperforms baseline approaches.
- TRIPS exhibits strong adaptability across diverse use cases.

Thank You!