FACT: Mitigating Inconsistent Hallucinations in LLMs via Fact-Driven Alternating Code-Text Training

Xinxin You[1] **Qixin Sun**[2] **Chenwei Yan**[3] **Xiao Zhang**[4] **Chen Ning**[1]
Xiangling Fu[5] **Si Liu**[2] **Guoping Hu**[6] **Shijin Wang**[6*] **Ji Wu**[1,7*] **Xien Liu**[1*]
[1]Department of Electronic Engineering, Tsinghua University, Beijing, China
[2]School of Artifcial Intelligence, Beihang University, Beijing, China
[3]School of Information Technology and Management,
University of International Business and Economics, Beijing, China [4]ByteDance, Beijing, China
[5]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
[6]iFLYTEK Research, Hefei, China [7]College of AI, Tsinghua University, Beijing, China
{yxx23,nc22}@mails.tsinghua.edu.cn {sunqx,liusi}@buaa.edu.cn {chenwei.yan@uibe.edu.cn}
{zhangxiao.12@bytedance.com} {fuxiangling@bupt.edu.cn} {gphu,sjwang3}@iflytek.com
{wuji_ee.xeliu}@mail.tsinghua.edu.cn

Abstract

Inconsistent hallucinations remain a major challenge for large language models (LLMs), undermining the accuracy and reliability of fact-based reasoning in real-world applications. Existing approaches often rely on task-specific training or adaptation, such as hand-crafted synthetic datasets for domain tasks or solutions mainly focused on numerical reasoning, thereby limiting generalizability to broader, unseen NLP tasks. Inspired by the structural rigor and logical consistency of programming languages, we observe that fact-based texts can be mapped to programming structures due to their inherent patterns. We further propose FACT, a novel **F**act-driven **A**lternating **C**ode-text **T**raining framework that alternates between text-to-code and code-to-text prediction. FACT is the first task-agnostic paradigm that embeds code and natural language in a shared semantic space, thereby transferring the logical consistency of code to LLM outputs in NLP tasks. Experiments show that with only a small subset of Wiki-40B-en for training, FACT reduces inconsistent hallucinations by 2.7%–8.0% and improves overall performance by 2.5%–6.1% in three leading LLMs and four diverse datasets covering QA and summarization tasks. This framework offers a new perspective on addressing challenging hallucinations in LLMs, contributing to more reliable AI.

1 Introduction

Achieving human-like logical consistency in fact-based reasoning is essential to advance artificial general intelligence (AGI) [1, 2]. Despite significant progress in large language models (LLMs) across various natural language processing (NLP) tasks [3, 4, 5], these models still frequently produce inconsistent hallucinations due to limited logical rigor [6, 7]. Such hallucinations occur mainly in two forms: (1) input-conflicting hallucinations, where the generated content diverges from the provided input [8, 9]; and (2) context-conflicting hallucinations, where the generated content contradicts information previously produced by the model [10, 11], as shown in the left panel of Fig. 1. These errors are particularly prevalent in fact-based reasoning, such as LLMs generating patient diagnoses in the medical domain that contradict clinical input data, or producing legal case summaries that

[*]Corresponding author

misrepresent the actual case facts. Such issues severely undermine the credibility of LLMs in real-world applications. Therefore, it is urgent to develop a universal approach that can effectively address these inconsistent hallucinations across a wide range of tasks.

Existing approaches for mitigating hallucinations in LLMs often require task-specific training or adaptation. Recent efforts have addressed inconsistencies by constructing domain-specific hand-crafted reasoning datasets—for instance, LOGIQA [12] for civil service exams and AR-LSAT [13], ReClor [14] for legal reasoning. However, such manual curation is time-consuming and lacks scalability. Consequently, synthetic data approaches have emerged, aiming to automate data generation and expand coverage. Nonetheless, these methods are primarily limited to numerical reasoning tasks, such as KPDDS [15], OpenMathInstruct-1 [16], and MathGenie [17], where large-scale question-answer pair construction is tractable. Attempts to address hallucinations in broader NLP tasks include methods such as Lookback Lens and SymbCoT [18, 19]. However, they still rely on significant task-specific adaptation, including separate classifiers or symbolic CoT pipelines designed for each task. As a result, while effective in narrow domains, these approaches generally lack generalizability and scalability to diverse, unseen NLP scenarios.

Advancements in training LLMs on large-scale code datasets, such as CoCoGen and CodeRL, have significantly reduced inconsistent hallucinations in code generation tasks [20, 21]. Building on this integration of code with LLMs, recent methods further improve logical rigor in reasoning tasks. For example, program-of-thought [22] and program-assisted language models [23] translate natural language problems into executable code and utilize code interpreters to derive answers. However, these methods face practical limitations: due to fundamental differences in organizational structure, modes of expression, and language style between code and natural language, only a narrow subset of problems, typically mathematical questions, can be effectively translated into executable code [22, 23]. Consequently, these approaches fail to transfer the logical consistency benefits of code to LLM outputs in a broader range of NLP tasks.

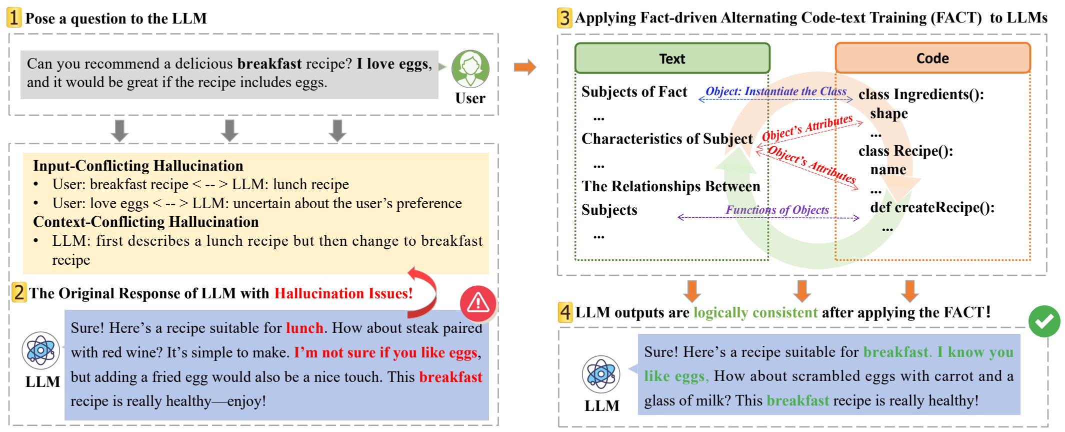
To address the above challenges, we propose FACT: mitigating inconsistent hallucinations in LLMs via **F**act-driven **A**lternating **C**ode-text **T**raining. FACT is grounded in the observation that fact-based texts exhibit intrinsic structural patterns that are amenable to systematic mapping onto programming abstractions. As illustrated in Fig. 1 (upper right), subjects and their attributes in natural language correspond to objects and properties in code, while processes and causal relationships are represented as functions. To further transfer the logical rigor of code to natural language, FACT implements an alternating training paradigm between text-to-code and code-to-text prediction. The key challenge—the absence of ground-truth code supervision for text-to-code prediction—is addressed by employing a pseudo-labeling with a two-step verification process: 1) confirming syntactic correctness; and 2) assessing semantic fidelity to the original text. Experiments show that training on just a small Wiki-40B-en [24] subset fundamentally improves LLM output consistency. Extensive evaluations with three state-of-the-art LLMs demonstrate that FACT significantly reduces inconsistent hallucinations and improves task performance on summarization and QA tasks, all without task-specific adaptation.

We summarize the contributions of our method as follows:

- We propose FACT, the first task-agnostic framework that significantly mitigates inconsistent hallucinations in LLMs through alternating code-text training. This framework offers new perspectives for developing LLMs that deliver logical consistency and accurate responses.
- We identify a clear correspondence between the structures of fact-based text and code. This finding not only supports our alternating training methodology, but also provides a foundation for future research bridging the gap between textual and coding modalities.
- Experiments show that with only a small subset of Wiki-40B-en [24] for training, FACT reduces inconsistent hallucinations by 2.7%–8.0% and improves overall performance by 2.5%–6.1% in three leading LLMs and four diverse datasets covering QA and summarization tasks.

2 Related Work

Data-Centric Approaches-Manual Bottlenecks and Narrow Scope Recent studies suggest that hallucinations in LLMs are often attributed to deficiencies in training data, underscoring the importance of high-quality reasoning datasets for improving consistency[25, 26, 27]. Thus, domain-specific, handcrafted benchmarks have been developed, such as LOGIQA[12] for civil service exams,


1 Pose a question to the LLM
User
2 The Original Response of LLM with Hallucination Issues!
LLM
3 Applying Fact-driven Alternating Code-text Training (FACT) to LLMs
Text
Code
Subjects of Fact
...
Characteristics of Subject
...
The Relationships Between Subjects
...
Object: Instantiate the Class
Object's Attributes
Object's Attributes
Functions of Objects
4 LLM outputs are logically consistent after applying the FACT!
LLM

Can you recommend a delicious **breakfast** recipe? **I love eggs**, and it would be great if the recipe includes eggs.

Input-Conflicting Hallucination
- User: breakfast recipe < -- > LLM: lunch recipe
- User: love eggs < -- > LLM: uncertain about the user's preference

Context-Conflicting Hallucination
- LLM: first describes a lunch recipe but then change to breakfast recipe

Sure! Here's a recipe suitable for **lunch**. How about steak paired with red wine? It's simple to make. **I'm not sure if you like eggs**, but adding a fried egg would also be a nice touch. This **breakfast** recipe is really healthy—enjoy!

```
class Ingredients():
    shape
    ...
class Recipe():
    name
    ...
    def createRecipe():
        ...
```

Sure! Here's a recipe suitable for **breakfast**. **I know you like eggs**. How about scrambled eggs with carrot and a glass of milk? This **breakfast** recipe is really healthy!

Figure 1: This figure illustrates that standard LLM responses (left) are susceptible to input-conflicting and context-conflicting hallucinations, resulting in outputs that are inconsistent with both user intent and contextual logic. The FACT approach (right) leverages structural alignment between factual text and code via fact-driven alternating code-text training, effectively reducing these hallucinations and producing responses that are both semantically and logically consistent with user intent and context.

AR-LSAT[13], and ReClor[14] for legal exams, though at considerable cost and labor. This has motivated interest in automated and synthetic dataset construction[28], particularly for numerical reasoning tasks where large-scale question-answer pairs are more easily generated (e.g., KPDDS[15], OpenMathInstruct-1[16], MathGenie[17]). However, these data-centric methods are generally tailored to specific domains or tasks, which limits their generalizability to broader NLP applications.

Symbol-Augmented CoT Approaches-Limited by Symbolic Expressivity Chain-of-Thought (CoT) strategies have shown promise in reducing hallucinations in reasoning tasks by promoting explicit step-by-step inference [29]. This line of work is supported by developing logical reasoning datasets such as LogiCoT [30] and FOLIO [31], which provide explicit reasoning chains. Furthermore, methods like LOGIC-LM[2] integrate symbolic solvers with language models to formalize natural language queries, while SymbCoT[19] enriches CoT prompting with symbolic expressions and logical rules. However, these methods are limited by symbolic expressivity and are less effective for complex problems, underscoring the need for more flexible and widely applicable solutions.

Code-Based Approaches-Limited Generalization Beyond Mathematics Recent research demonstrates that training LLMs on code datasets helps reduce hallucinations in code generation by leveraging the strict syntax and semantics of programming languages [20, 21]. Building on this integration of code with LLMs, recent methods further improve logical rigor in reasoning tasks. Approaches like Program of Thought (PoT) [22] and Program-Aided Language Models (PAL) [23] translate natural language problems into executable code and utilize code interpreters to derive answers. However, fundamental differences between code and natural language restrict these methods to mainly mathematical problems, limiting the transfer of code's logical consistency benefits to LLM outputs in a broader range of NLP tasks.

3 Method

Our approach builds on the observation that fact-based texts possess inherent structural properties, which can be systematically mapped to programming abstractions, as illustrated in Fig. 1 (upper right). To leverage this, we propose an alternating training paradigm between text-to-code and code-to-text generation. Due to the absence of ground-truth code for text-based code prediction, we generate pseudo-labels to supervise model training. To ensure the quality of the generated code, we apply a two-stage assessment that verifies syntactic correctness and evaluates semantic fidelity relative to the original text. This training process aligns text and code representations within a unified semantic space, effectively transferring the logical consistency of code to LLM outputs on broader NLP tasks.

The remainder of this section is organized as follows. Section 3.1 formalizes the problem setting and outlines our research objectives. Section 3.2 introduces a procedure for selecting fact-based texts.

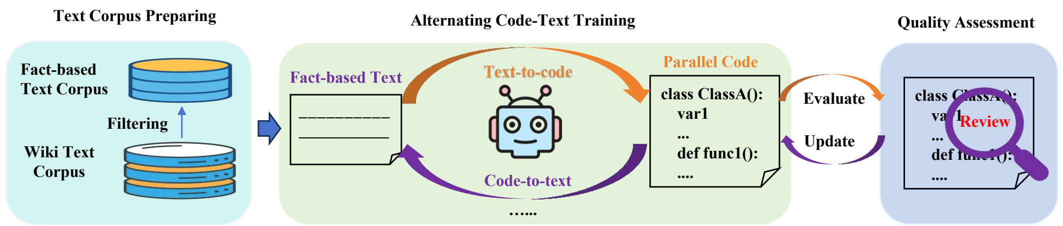


Figure 2: An overview of FACT begins with the filtering of fact-based text, followed by alternating generation training of fact-based text and parallel code based on their transformation relationship. In each iteration, a quality evaluation module is employed to assess the quality of the generated parallel code, ultimately achieving semantic and stylistic alignment between natural language and code.

Section 3.3 details the alternating code-text training mechanism. Section 3.4 describes the proposed quality assessment and adaptive loss formulation.

3.1 Problem Definition

We consider the task of text generation with a large language model (LLM). Given an external prompt p that specifies the generation objective and an input context t providing background information, the model generates an output sequence of sentences $\mathbf{x} = [x_1, x_2, \ldots, x_{|\mathbf{x}|}]$, where each x_i is a sentence and $|\mathbf{x}|$ denotes the sequence length. The overall generation process can be formulated as $\mathbf{x} \sim \mathrm{LLM}(\cdot \mid p, t)$. A central challenge in LLM-based generation is ensuring contextual and internal consistency. We focus on two types of hallucination errors that may arise in the outputs:

Input-Conflicting Hallucination This refers to the generated sequence $\mathbf{x}$ contains information that contradicts the input t. Formally, we define input-conflicting hallucination as:

$$H(\mathbf{x}, t) = \begin{cases} 1, & \exists\, \mathbf{x} \neq t \ \text{ and } \ \mathrm{Contradict}(\mathbf{x}, t) \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Context-Conflicting Hallucination. This type of inconsistency arises when any two sentences within the generated output $\mathbf{x}$ are logically contradictory, even though they are generated under the same prompt and context. Formally, we define:

$$H(x_i, x_j) = \begin{cases} 1, & \exists\, i \neq j \ \text{ and } \ \mathrm{Contradict}(x_i, x_j) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathrm{Contradict}(x_i, x_j)$ denotes that the i-th and j-th sentences in $\mathbf{x}$ are logically inconsistent. By formally defining these types of hallucination, our framework aims to enforce both input consistency and internal coherence in LLM-generated outputs.

3.2 Text Filtering for Implicit Facts

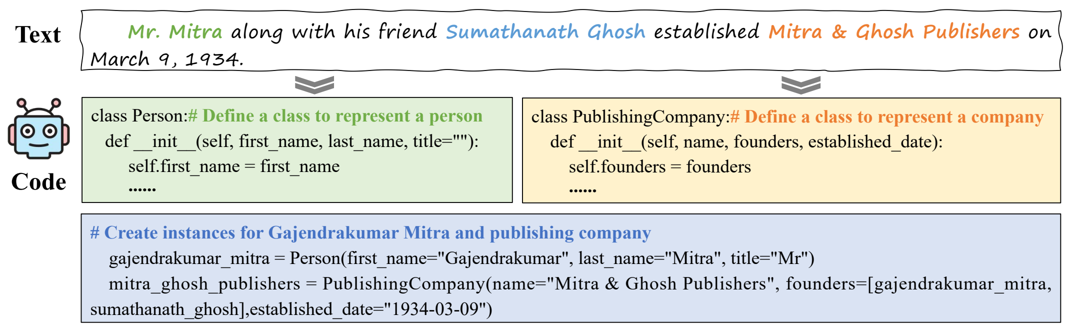
To support effective alternating training, we filter input texts to retain only those containing factual content suitable for structured code representation. This is achieved via an LLM-based filtering mechanism E_{llm}, which determines whether a text t meets the criteria. Specifically, we use GPT-3.5 with a tailored prompt (see Appendix A.1) to perform this assessment. The decision rule is

$$D(t) = \begin{cases} 1, & \text{if } E_{\mathrm{llm}}(t) = \mathrm{True}, \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $D(t) = 1$ means t is selected; otherwise, it is discarded. Rigorous text filtering provides a solid foundation for the effectiveness of subsequent alternating code-text generation training.

3.3 Alternating Code-Text Training

Predicting code from text and reconstructing text from generated code constitute the core of our training framework, as illustrated in Fig. 2. We adopt an alternating training paradigm between



Mr. Mitra along with his friend Sumathanath Ghosh established Mitra & Ghosh Publishers on March 9, 1934.

```
class Person:# Define a class to represent a person
  def __init__(self, first_name, last_name, title=""):
     self.first_name = first_name
     ......
```

```
class PublishingCompany:# Define a class to represent a company
  def __init__(self, name, founders, established_date):
     self.founders = founders
     ......
```

```
# Create instances for Gajendrakumar Mitra and publishing company
  gajendrakumar_mitra = Person(first_name="Gajendrakumar", last_name="Mitra", title="Mr")
  mitra_ghosh_publishers = PublishingCompany(name="Mitra & Ghosh Publishers", founders=[gajendrakumar_mitra,
sumathanath_ghosh],established_date="1934-03-09")
```

Figure 3: A simplified example showing a text segment and its corresponding code. The code first defines classes—Person and PublishingCompany—to capture the roles and structure described in the text, and then instantiates them to represent the co-founders and the company. Only a single sentence is shown here due to space constraints; another complete example is provided in Appendix C.

text-to-code and code-to-text generation, promoting the alignment of both modalities within a unified semantic space. Both directions are formulated as supervised fine-tuning (SFT) tasks.

Text-to-Code with Pseudo-Label Supervision In the text-to-code direction, since ground-truth code is unavailable, we employ pseudo-label self-supervision. For each input t, the LLM G with prompt p_{t2c} (see Appendix A.2) generates a pseudo-labeled code sequence, which serves as the supervision signal for training. During training, given the same input t and prompt, the model predicts the code sequence as

$$c \sim G(\cdot \mid p_{t2c}, t). \tag{4}$$

Both the pseudo-labeled code and the predicted code are produced using the same exemplar-based prompt to ensure consistency and alignment with the input semantics.

Code-to-Text Supervision In the code-to-text direction, given the generated code sequence c and the prompt p_{c2t} (see Appendix A.3), the model reconstructs the original fact-centric text:

$$t \sim G(\cdot \mid p_{c2t}, c). \tag{5}$$

The original text is used as the ground-truth target during training. For illustration, Fig.3 presents a simplified example of both the original text and its corresponding code due to space limitations. A complete example is provided in Appendix C.

3.4 Quality Assessment and Adaptive Loss

While the alternating code-text training framework enables mutual supervision, it remains critical to ensure that generated code is not only syntactically correct but also semantically faithful to the source text. To address this, we introduce a two-stage quality assessment and an adaptive loss objective, explicitly incorporating both syntactic and semantic feedback into the training process.

(1) Syntactic Validity Each generated code c_i is executed by a Python interpreter to ensure syntactic correctness. Only code that executes successfully—i.e., free of syntax errors, undefined variables, and incomplete constructs—is retained for subsequent evaluation. For any code that fails the syntactic validity check, we assign a fixed similarity score of $S_i = 0.1$ (see Appendix B.1 for details) in the subsequent loss computation.

(2) Semantic Fidelity via Reverse Reconstruction To assess whether the generated code c_i captures the intended semantics of the original text t_i, we conduct a reverse reconstruction. With a tailored prompt p_{reorg} (see Appendix A.4), let model generates a reorganized template c_i^{reorg} by replacing information such as entities and attributes with code variables from c_i:

$$c_i^{\text{reorg}} \sim G(\cdot \mid p_{\text{reorg}}, t_i) \tag{6}$$

By executing the generated code c_i and substituting runtime values into the placeholders of c_i^{reorg}, we obtain the instantiated text $c_i^{\text{reorg}'}$. To evaluate semantic fidelity between the generated and original texts, we employ a composite metric combining ROUGE-1 and ROUGE-L [32], where ROUGE-1 captures factual entities, and ROUGE-L reflects global structural similarity. The overall similarity score is defined as:

$$S_i = \frac{1}{2}\left(\text{ROUGE-}L(c_i^{\text{reorg}'}, t_i) + \text{ROUGE-1}(c_i^{\text{reorg}'}, t_i)\right) \tag{7}$$

A higher value of S_i indicates a greater degree of semantic fidelity of the code to the original text, as the instantiated entities and related content more closely match those described in the original. For clarity, based on the example provided in Fig. 3, we show that the reorganized text c_i^{reorg} is composed of references to code entities and properties:

mitra_ghosh_publishers.founders[0].title mitra_ghosh_publishers.founders[0].last_name along with his friend mitra_ghosh_publishers.founders[1].first_name mitra_ghosh_publishers.founders[1].last_name established mitra_ghosh_publishers.name on mitra_ghosh_publishers.established_date.

Upon execution of c_i, every placeholder in c_i^{reorg} is programmatically replaced with its run-time value, resulting in the fully instantiated text $c_i^{\text{reorg}'}$ (as illustrated, all placeholders are correctly replaced):

Mr Mitra along with his friend Sumathanath Ghosh established Mitra & Ghosh Publishers on March 9, 1934.

Adaptive Loss For each training example, we weight the text-to-code loss by $(1 - S_i)$, so that samples with lower quality have a larger penalty on model updates. For a batch size n, the final loss of the text-to-code generation process for each iteration is defined as:

$$\mathcal{L}_{\text{t2c}} = \frac{1}{n}\sum_{i=1}^{n}(1 - S_i)\,\mathcal{L}_{i,\text{t2c}} \tag{8}$$

where $\mathcal{L}_{i,\text{t2c}}$ denotes the loss of the text-to-code generation process for sample i in this iteration.

In our framework, pseudo-code labels are iteratively regenerated at each training round using the latest model checkpoint, rather than being generated only once. This enables the supervision signal to dynamically improve alongside the model's evolving capabilities, yielding progressively higher-quality pseudo-labels that further enhance modality alignment. Through this alternating code-text training process, our approach establishes a robust and generalizable framework that consistently improves logical consistency and effectively mitigates inconsistent hallucinations of LLMs.

4 Experiment

4.1 Settings

Model We evaluated three base models: LLaMA-3.1-Instruct-8B [28], Ministral-Instruct-8B [33], and Qwen-2.5-Instruct-7B [34]. These recent and widely used instruction-tuned LLMs have comparable parameter sizes (7B/8B) and demonstrate strong performance across various NLP tasks. All models are publicly available for reproducibility.

Datasets We randomly sampled 10,000 entries from the Wiki-40B-en[24] dataset, using only the first paragraph of each entry. After fact-based filtering, 53.74% (5,374) were retained for alternating training, while 27.85% and 18.41% were labeled as non-factual and invalid (see 4.3). Owing to our alternating approach (text-to-code and code-to-text), each sample is used bidirectionally, yielding 10,748 samples (5,374 $\times$ 2) per iteration.

For evaluation, we selected benchmark datasets for both text summarization and question answering (QA). Specifically, we used CNN/Daily Mail[35] and SAMSum[36] for summarization, and SQuAD v2[37, 38] together with HaluEval[39] for QA. These datasets were chosen to comprehensively cover both classic and emerging scenarios: CNN/Daily Mail and SQuAD v2 are widely recognized classics, while SAMSum provides a high-quality, human-annotated benchmark for dialogue summarization, and HaluEval (introduced in 2023) specifically targets hallucination evaluation in LLMs. Since HaluEval only provides the construction methodology, we built the dataset ourselves following its

Table 1: Hallucination evaluation results for all methods and backbone models across datasets are reported. "Consis" and "AlignS" denote Consistency and AlignScore, respectively.

Method	Summary Task				QA Task			
	CNN/Daily Mail		SAMSum		SQuAD V2		HaluEval	
	Consis(↑)	AlignS(↑)	Consis(↑)	AlignS(↑)	Anah-v2(↓)	AlignS(↑)	Anah-v2(↓)	AlignS(↑)
LLaMA-3.1-Instruct-8B								
Base	86.40	83.28	90.79	90.67	14.32	95.94	12.48	96.35
Prompt	87.73	86.51	91.08	90.98	15.02	95.50	8.92	96.17
SFT	90.33	84.06	89.56	87.14	13.14	95.24	8.27	98.52
SymbCoT	86.80	84.28	90.23	88.37	13.82	95.79	11.66	96.58
Lookback	83.12	81.75	81.12	84.05	17.62	93.08	14.31	92.82
FACT	**91.07**	**87.29**	**92.77**	**91.58**	**8.39**	**98.33**	**7.01**	**98.93**
Ministral-Instruct-8B								
Base	87.30	82.35	89.14	88.23	13.31	79.15	12.15	94.16
Prompt	88.02	83.12	90.92	89.10	14.53	95.64	13.02	95.11
SFT	90.10	83.54	89.45	87.33	12.65	86.47	10.36	96.61
SymbCoT	88.26	82.10	84.98	85.47	16.65	95.60	10.60	96.14
Lookback	84.37	80.15	82.13	85.95	15.40	92.11	14.26	93.04
FACT	**90.93**	**86.21**	**93.36**	**92.63**	**9.12**	**97.75**	**7.64**	**98.84**
Qwen-2.5-Instruct-7B								
Base	83.76	82.56	85.23	86.42	17.22	93.65	14.75	95.05
Prompt	84.51	84.83	86.06	85.41	16.31	93.93	13.98	95.14
SFT	89.12	81.13	87.69	88.95	12.41	94.17	11.14	96.39
SymbCoT	84.45	81.84	84.31	86.72	13.25	93.67	11.38	95.35
Lookback	83.84	81.52	81.25	83.30	16.18	92.19	14.36	93.17
FACT	**90.70**	**85.53**	**89.77**	**90.21**	**8.22**	**98.56**	**9.22**	**97.67**

original paper. For fair comparison, we also randomly sampled 10,748 instances from each dataset for training the baseline methods.

Baselines We compare FACT with the following baselines, covering diverse modeling paradigms: (1) Base Model: the original instruction-tuned model; (2) Naive Prompting: the model was prompted with straightforward instructions (see Appendix A.5); (3) Supervised Fine-Tuning (SFT): model fine-tuned on task-specific datasets; (4) SymbCoT [19]: a recent method that reformulates chain-of-thought reasoning as symbolic inference to improve faithfulness; (5) Lookback [18]: a hallucination detection and mitigation framework using a linear classifier on lookback ratio features.

Metrics For hallucination evaluation, we adopt several recently proposed targeted metrics: AlignScore [40] for both summarization and QA, UniEval Consistency [41] specifically for summarization, and Anah-V2 [42] exclusively for QA. We also report overall performance metrics, including Coherence and Relevance [41] for summarization, and F1 as well as Exact Match (EM) [43] for QA.

Implementation Details To ensure fair comparison, all models were trained and evaluated under identical configurations. For the Base Model and Prompting variants of each backbone, inference was conducted using LLaMA-Factory[2] on four NVIDIA GeForce RTX 4090 GPUs (24GB each); both models, being instruction-tuned, adopted default chat templates and greedy decoding. FACT and SFT were also trained on the same four RTX 4090 GPUs with LLaMA-Factory for three epochs, with a learning rate of 1×10^{-4}, batch size 32, and LoRA (rank 8) for consistent acceleration. During inference, these models also employed greedy decoding. For the SymbCoT and Lookback baselines, we used the official implementations and configurations, replacing their backbone models with LLaMA, Mistral, and Qwen.

4.2 Evaluation Results

Hallucination Evaluation Results As shown in Table 1, FACT consistently outperforms all baselines across all models and datasets in hallucination metrics. For instance, with the LLaMA backbone, FACT achieves average improvements of 4.19% (Consistency) and 3.31% (AlignScore) on CNN/Daily Mail, 4.21% (Consistency) and 3.34% (AlignScore) on SAMSum, 6.39% (Anah-v2) and 3.22% (AlignScore) on SQuAD v2, and 4.12% (Anah-v2) and 2.84% (AlignScore) on HaluEval.

[2]`https://github.com/hiyouga/LLaMA-Factory`

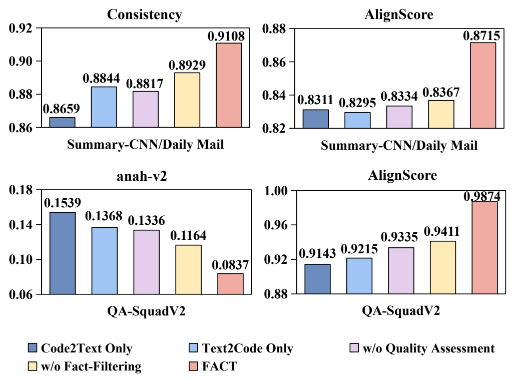


Figure 4: Ablation results on CNN/Daily Mail and SQuADv2 with the LLaMA backbone, illustrating inconsistent hallucination metrics.

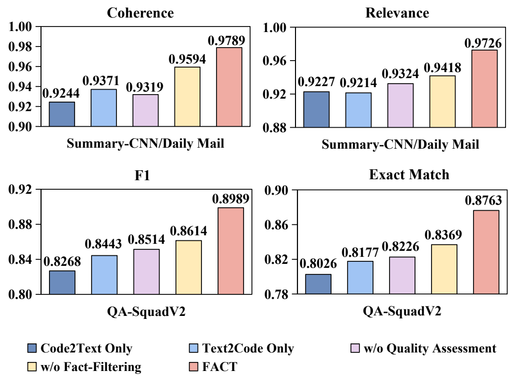


Figure 5: Ablation results on CNN/Daily Mail and SQuADv2 with the LLaMA backbone, illustrating task-specific evaluation metrics.

Beyond horizontal comparisons across datasets, vertical comparisons against individual baselines on LLaMA further highlight the superiority of FACT, with average improvements of 7.57% over Lookback, 3.95% over SymbCoT, 3.49% over Base, 2.64% over SFT, and 2.57% over Prompt across all tasks. Similar results for Ministral and Qwen further highlight the robustness and generalizability of FACT. The largest margin is against Lookback, which relies on hallucination-annotated data and is thus sensitive to annotation quality, while SymbCoT, though strong for first-order logic symbolic reasoning, is less effective on tasks with complex relationships. Overall, these findings establish our fact-driven alternating code-text training as a novel framework that effectively transfers code's logical rigor to LLMs' outputs, enabling reliable and accurate reasoning across a wide range of NLP tasks.

Task Performance Evaluation Results As shown in Table 2, FACT also delivers consistent improvements on both summarization and QA task metrics. For instance, with LLaMA, FACT achieves average improvements of 6.35% over Base, 3.81% over Prompt, 6.76% over SFT, 2.64% over SymbCoT, and 5.19% over Lookback across all tasks. Comparable gains are observed with Ministrel and Qwen, supporting the robustness of the conclusions. These experimental results demonstrate that enhancing consistency substantially boosts overall task performance. This improvement can be attributed to the prevention of the "hallucination snowballing" phenomenon [44], where hallucinations accumulate during generation, resulting in compounding errors and degraded output quality. By applying FACT to LLMs, we curb the accumulation of such errors, ultimately promoting the generation of more accurate outputs and thereby improving overall task performance.

Ablation Study Results We conduct comprehensive ablation studies on the CNN/Daily Mail and SQuAD V2 datasets using the LLaMA backbone to systematically evaluate the contribution of each FACT module. Specifically, we examine the effects of omitting the factual text filter, disabling alternating training by using only text-to-code or only code-to-text generation, and removing the quality assessment module. **For hallucination evaluation,** as shown in Fig. 4, retaining only the code-to-text pathway results in the largest increase in hallucination rate (5.72% on average), followed by using only text-to-code (4.69%), removing the quality assessment module (4.28%), and omitting the factual text filter (3.29%). Similar patterns are observed for overall **task performance**: using only code-to-text, only text-to-code, removing the quality assessment, and omitting the factual text filter lead to mean performance drops of 6.23%, 5.13%, 4.69%, and 3.16%, respectively (see Fig. 5). These results underscore the necessity of including all modules in the FACT design. Among them, the text-to-code pathway within the alternating training scheme is particularly valuable, as it guides the model to parse and organize factual content into code-like, logically consistent representations, thereby strengthening alignment between language and structured information. In comparison, the code-to-text pathway mainly helps the model retain the ability to generate fluent and semantically coherent text.

Table 2: Task evaluation results for all methods and backbone models across datasets.

Method	Summary Task				QA Task			
	CNN/Daily Mail		SAMSum		SQuAD V2		HaluEval	
	Coherence	Relevance	Coherence	Relevance	F1	Exact Match	F1	Exact Match
LLaMA-3.1-Instruct-8B								
Base	92.40	92.61	93.25	90.33	83.59	79.07	81.33	77.46
Prompt	96.82	95.56	94.84	93.17	84.37	80.21	84.69	80.52
SFT	90.56	88.29	91.49	89.67	82.29	81.25	83.79	79.25
SymbCoT	96.52	95.47	96.37	95.78	82.14	86.20	85.44	81.73
Lookback	94.31	90.06	93.62	92.74	81.63	80.81	83.69	82.25
FACT	**97.89**	**97.26**	**98.77**	**98.09**	**88.89**	**87.63**	**87.26**	**84.86**
Ministral-Instruct-8B								
Base	92.73	92.32	93.32	92.67	84.65	83.31	80.22	77.54
Prompt	94.26	92.87	94.97	93.65	85.59	81.78	82.49	78.20
SFT	91.04	90.33	90.25	90.33	81.77	80.42	82.86	81.53
SymbCoT	95.61	94.88	92.34	91.60	84.06	82.79	84.98	81.72
Lookback	93.32	92.15	91.29	90.06	82.44	80.51	83.65	82.77
FACT	**96.47**	**96.16**	**96.28**	**95.39**	**88.17**	**85.36**	**86.22**	**84.13**
Qwen-2.5-Instruct-7B								
Base	89.33	88.45	88.26	87.34	80.14	78.25	78.56	76.47
Prompt	92.61	92.18	90.14	89.12	80.59	79.31	81.49	81.05
SFT	93.38	93.48	90.51	91.08	81.44	80.52	80.30	79.25
SymbCT	94.71	93.25	92.25	91.83	82.17	80.44	82.57	81.02
Lookback	94.89	93.62	91.34	91.58	80.44	79.16	81.19	80.44
FACT	**95.50**	**95.14**	**95.72**	**94.04**	**86.59**	**84.72**	**84.34**	**83.28**

4.3 Analysis and Discussion

Analysis of the Text Filter Module After fact-based filtering of 10,000 Wiki-40B-en entries as described earlier, 53.74% were retained as fact-type texts, 27.85% were classified as non-fact-type, and 18.41% were labeled as invalid, largely due to unclear responses from the LLM. To further assess the reliability of the filtering process, three engineers independently annotated 100 samples each of fact-type and non-fact-type texts. Based on majority voting, the filter achieved a true positive rate of 93% and a false negative rate of 7% for fact-type texts, as well as a true negative rate of 92% and a false positive rate of 8% for non-fact-type texts. Additionally, a manual review of 100 invalid samples confirmed that 96% genuinely lacked valid labels, primarily because the model failed to follow instructions, generated responses in unexpected formats, or was unable to judge ambiguous inputs. These results demonstrate the reliability of the fact-driven filtering process, which provides a robust data foundation for subsequent alternating code-text training.

Code Generation Quality under Pseudo-Labels To assess whether our pseudo-labeling with two-stage quality assessment compensates for the absence of ground-truth code, we monitor the quality of generated code across training iterations. Effective supervision should manifest as a general improvement. As shown by the purple line in the upper panel of Fig. 6, the proportion of executable code starts at 87.28%—attributable to rigorous factual text filtering—and generally increases over epochs, despite minor fluctuations, indicating enhanced syntactic validity. Similarly, semantic fidelity, measured by the similarity between reconstructed and original text, rises by up to 5.23% (red line, lower panel) before plateauing after the third iteration, which motivates our three-iterations schedule. Overall, these results show that our approach not only enables effective training without gold-standard code, but also creates a virtuous cycle in which the model's code generation ability and overall performance reinforce each other across successive iterations. Further analysis of failure cases after three iterations is provided in Appendix B.2.

Effectiveness of Structured Code-based Inference To assess the effectiveness of structured code as an intermediate representation, we perform an ablation study using a two-stage inference pipeline: source texts are first converted into structured code (with unconvertible segments retained as plain text), and summaries or answers are then generated from the converted code. On the CNN/Daily Mail dataset with a LLaMA backbone, code-based inference achieves 90.24% Consistency (vs. end-to-end 91.08%, see Table 1), 86.44% AlignScore (vs. 87.15%), 97.01% Coherence (vs. 97.89%), and

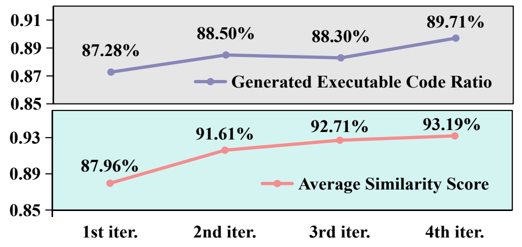


Figure 6: The trends of the proportion of generated executable code and the similarity score between reconstructed and original text as the number of training iterations increases.

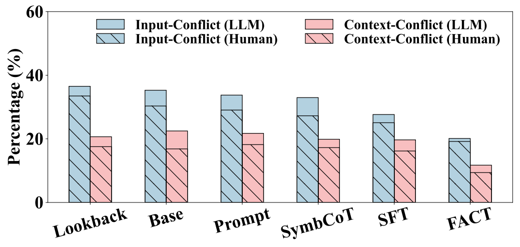


Figure 7: Sentence-level hallucination rates for input-conflicting and context-conflicting hallucinations, evaluated using the GPT-4.1 and human across different methods.

96.72% Relevance (vs. 97.26%). On SQuAD V2, our pipeline yields 9.54% on Anah-v2 (vs. 8.37%), 96.81% on AlignScore (vs. 98.84%), 87.73% F1 (vs. 88.89%), and 87.05% EM (vs. 87.63%). These minimal reductions in consistency and task performance demonstrate that structured code serves as a transparent and faithful alternative representation, preserving essential semantic information for downstream tasks. This provides strong evidence for the effectiveness of the FACT framework in aligning text and code within a unified semantic space and enabling high-quality code representations.

Analysis of Input- and Context-Conflicting Hallucinations We evaluated FACT on challenging samples from the CNN/Daily Mail dataset, specifically selecting cases from base LLaMA outputs with consistency scores below 0.6 to target strong logical inconsistencies. Evaluation was performed both automatically, using the powerful GPT-4.1 with the prompt in Appendix A.6, and manually, by averaging the assessments of three engineers on 100 outputs randomly sampled from the above-selected cases. We quantified hallucination by measuring the proportion of sentences showing input- and context-conflicting errors. As shown in Fig. 7, FACT substantially reduces hallucination rates compared to other methods. Relative to SFT (the strongest baseline), GPT-based evaluation indicates reductions of 7.53% in input-conflicts and 5.89% in context-conflicts. Manual assessment yields similar decreases of 7.96% and 3.79%, highlighting the robustness of our findings. Additionally, context conflicts are generally less frequent than input conflicts, and FACT achieves especially notable improvements in addressing this type of hallucination. This may be attributed to the text-to-code prediction stage introduced during FACT training, where the process of generating code encourages the model to maintain higher-quality and more consistent contextual logic.

5 Conclusion

In this work, we present FACT, a novel, task-agnostic framework that mitigates inconsistent hallucinations in LLMs through fact-driven alternating code-text training. By unifying fact-based text and code representations in a shared semantic space, FACT brings the logical rigor and consistency of code to NLP outputs. Experiments on three leading LLMs and diverse benchmarks demonstrate that FACT greatly reduces hallucinations and consistently improves task performance. This framework not only establishes a new paradigm for addressing persistent hallucination challenges in LLMs, but also lays a solid foundation for more trustworthy and robust deployment of LLMs in real-world, fact-intensive reasoning scenarios.

Limitations

Despite the demonstrated effectiveness of our approach, several issues remain in the generated code that warrant further attention. Some outputs still omit essential elements, such as key methods or complete attribute definitions. In addition, while overall logical consistency is generally preserved, certain complex cases exhibit incomplete handling of dependencies or lack fine-grained organization. There are also cases with class designs that do not fully represent the intended structure or relationships, as well as expressions that lack clarity or contain formatting inconsistencies. Addressing these issues—by improving content coverage, structural accuracy, and code clarity—may further enhance the robustness of FACT in future work.

Acknowledgments

The work is supported by the Noncommunicable Chronic Diseases-National Science and Technology Major Project (Grant No.2023ZD0506501) and the Beijing Natural Science Foundation (No. 4252046). We thank the anonymous reviewers for helpful comments and feedback.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, 2023.

[3] Xuhui Jiang, Yuxing Tian, Fengrui Hua, Chengjin Xu, Yuanzhuo Wang, and Jian Guo. A survey on large language model hallucination via a creativity perspective. *arXiv preprint arXiv:2402.06647*, 2024.

[4] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. In *The Twelfth International Conference on Learning Representations*.

[5] Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henghui Zhu, Rui Dong, Deguang Kong, Juliette Burger, Anjelica Ramos, William Yang Wang, George Karypis, et al. Street: A multi-task structured reasoning and explanation benchmark. In *The Eleventh International Conference on Learning Representations*.

[6] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.

[7] Forrest Sheng Bao, Miaoran Li, Renyi Qu, Ge Luo, Erana Wan, Yujia Tang, Weisi Fan, Manveer Singh Tamber, Suleman Kazi, Vivek Sourabh, et al. Faithbench: A diverse hallucination benchmark for summarization by modern llms. *arXiv preprint arXiv:2410.13210*, 2024.

[8] David Dale, Elena Voita, Loic Barrault, and Marta R Costa-jussà. Detecting and mitigating hallucinations in machine translation: Model internal workings alone do well, sentence similarity even better. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[9] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, 2021.

[10] Xiao Pu, Mingqi Gao, and Xiaojun Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023.

[11] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[12] Hanmeng Liu, Jian Liu, Leyang Cui, Zhiyang Teng, Nan Duan, Ming Zhou, and Yue Zhang. Logiqa 2.0—an improved dataset for logical reasoning in natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[13] Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*, 2021.

[14] Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

[15] Yiming Huang, Xiao Liu, Yeyun Gong, Zhibin Gou, Yelong Shen, Nan Duan, and Weizhu Chen. Key-point-driven data synthesis with its enhancement on mathematical reasoning. *arXiv preprint arXiv:2403.02333*, 2024.

[16] Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*, 2024.

[17] Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. *arXiv preprint arXiv:2402.16352*, 2024.

[18] Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1419–1436, 2024.

[19] Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*, 2024.

[20] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, 2022.

[21] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

[22] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

[23] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[24] Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40b: Multilingual language model dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2440–2452, 2020.

[25] Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. Rulebert: Teaching soft rules to pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1460–1476, 2021.

[26] Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, 2021.

[27] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, 2020.

[28] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[30] Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. Logicot: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921, 2023.

[31] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, et al. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

[32] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

[33] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[34] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[35] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[36] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70, 2019.

[37] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[38] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[39] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. *arXiv preprint arXiv:2305.11747*, 2023.

[40] Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. Alignscore: Evaluating factual consistency with a unified alignment function. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[41] Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. Towards a unified multi-dimensional evaluator for text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2023–2038, 2022.

[42] Yuzhe Gu, Ziwei Ji, Wenwei Zhang, Chengqi Lyu, Dahua Lin, and Kai Chen. Anah-v2: Scaling analytical hallucination annotation of large language models. *arXiv preprint arXiv:2407.04693*, 2024.

[43] Kevin Fischer, Darren Fürst, Sebastian Steindl, Jakob Lindner, and Ulrich Schäfer. Question: How do large language models perform on the question answering tasks? answer. *arXiv preprint arXiv:2412.12893*, 2024.

[44] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. In *Forty-first International Conference on Machine Learning*.

Appendix and Supplemental Material

A Introduction to the Prompt Used

A.1 The Prompt for Filtering Texts

Your task is to determine whether a text is suitable for description in code (e.g., modeling a relationship using class-based instances, modeling a process using a function), and return "yes" if it is suitable, or "no" otherwise. Generally speaking, an organized text that describes a series of facts and processes is suitable for code description, while a text that mainly expresses emotions or illogicality is not suitable for code description. The following is an example:
Input : The Hullaballoos were created in August 1964, but had been working in the UK for over three years under the name of Ricky Knight and The Crusaders. They were not named after the American television programme Hullabaloo. Their name came from the city of Hull, England, whence they hailed.
Output : yes
Please decide whether the following text is suitable for code description according to the above requirements:
FK Drezga is founded on 1972, as a team from the Piperi region near Podgorica. At their first season, FK Drezga played in Fourth League - Central region (the lowest rank in SFR Yugoslavia). Club was dissolved at the end of the seventies.

A.2 The Prompt for Predicting Code from Text

The prompt provided to the trained base model for alternating training in predicting code from text.

Please use a code snippet to encapsulate the given text into a structured format using classes and instances in Python. Here is an example for you.
Input:for Gajendrakumar Mitra: Mr Mitra along with his friend Sumathanath Ghosh established Mitra & Ghosh Publishers on March 9, 1934.
output:

```
#Define a class to represent a person
class Person:
    def __init__(self, first_name, last_name, title=""):
        self.first_name = first_name
        self.last_name = last_name
        self.title = title
#Define a class to represent a publishing company
class PublishingCompany:
    def __init__(self, name, founders, established_date):
        self.name = name
        self.founders = founders
        self.established_date = established_date
# Create instances for Gajendrakumar Mitra and Sumathanath Ghosh
gajendrakumar_mitra = Person(first_name="Gajendrakumar", last_name="Mitra", title="Mr")
sumathanath_ghosh = Person(first_name="Sumathanath", last_name="Ghosh")
# Create an instance for the publishing company Mitra & Ghosh Publishers
mitra_ghosh_publishers = PublishingCompany( name="Mitra & Ghosh", Publishers
founders=[gajendrakumar_mitra,sumathanath_ghosh], established_date="1934-03-09")
```

Please complete an encapsulation of the following text based on the above requirements and particles.
Bacon was born in Ipswich and lived in Great Yarmouth as a child. Bacon attended Yarmouth Art School from 1917-1923, where she won a scholarship in 1917 and by 1921 passed the Board of Education drawing examinations at the earliest age possible. She studied at the Norwich School of Art and then at the Royal College of Art in London, obtaining her diploma in 1927.

A.3 The Prompt for Predicting Text from Code

The prompt provided to the trained base model for alternating training in predicting text from code.

Here is a code snippet which encapsulates an original text into a structured format using classes and instances in Python. You are going to predict the original text after reading the code

```
#Define a class to represent a person
class Person:
    def __init__(self, first_name, last_name, title=""):
        self.first_name = first_name
        self.last_name = last_name
        self.title = title
#Define a class to represent a publishing company
class PublishingCompany:
    def __init__(self, name, founders, established_date):
        self.name = name
        self.founders = founders
        self.established_date = established_date
# Create instances for Gajendrakumar Mitra and Sumathanath Ghosh
gajendrakumar_mitra = Person(first_name="Gajendrakumar", last_name="Mitra", title="Mr")
sumathanath_ghosh = Person(first_name="Sumathanath", last_name="Ghosh")
# Create an instance for the publishing company Mitra & Ghosh Publishers
mitra_ghosh_publishers = PublishingCompany( name="Mitra & Ghosh", Publishers
founders=[gajendrakumar_mitra,sumathanath_ghosh], established_date="1934-03-09")
```

A.4 The Prompt for Predicting Reorganized Text

The prompt provided to the trained base model to predict reorganized text based on the generated code.

Please further utilize the original text and the generated code data to represent the entities in the original text with code, forming a new mixed text. Below is an example.

```
text=f"mitra_ghosh_publishers.founders[0].title
mitra_ghosh_publishers.founders[0].first_name
mitra_ghosh_publishers.founders[0].last_name along with his friend
mitra_ghosh_publishers.founders[1].first_name mitra_ghosh_publishers.founders[1].last_name established
mitra_ghosh_publishers.name on
mitra_ghosh_publishers.established_date.
```

original text : The Hullaballoos were created in August 1964, but had been working in the UK for over three years under the name of Ricky Knight and The Crusaders. They were not named after the American television programme Hullabaloo. Their name came from the city of Hull, England, whence they hailed.

A.5 The Prompt for the Base and Prompt Versions Used for Model Testing

Table 3: For the summary and QA tasks, show the instructions for the base and prompt versions of the three model bases.

Tasks	Base	Prompt
Summary	Write a summary of the following news.	Write a summary of the following news. Attention should be paid to the consistency of the abstract with the original text to avoid generating content with hallucinations.
QA	You are a question answerer. You should answer the questions directly based on the given reference without adding any prefixes or suffixes, and without analyzing the answers. After answering the question, do not say anything else. Reference document: ...... Please answer the question based on the above reference:	You are a question answerer. You should answer the questions directly based on the given reference without adding any prefixes or suffixes, and without analyzing the answers. After answering the question, do not say anything else. Please do not output content that is inconsistent with the context, and avoid giving irrelevant or contradictory answers. Reference document: ...... Please answer the question based on the above reference:

A.6 The Prompt for Evaluating Two Types of Inconsistent Hallucinations

Table 4: For the summary and QA tasks, provide the instructions for evaluating two types of hallucinations.

Tasks	**Input-Conflict**	**Context-Conflict**
Summary	The following presents data where a large language model generates summaries based on the story content, including the original story and the model's output. Please check whether the output contains any inconsistencies with the story, such as fabricated information not present in the original story. If inconsistencies are found, return all sentences containing them; if there are none, respond with "None."	The following content is the output of a large language model. Please check for any self-contradictory parts, specifically instances where the information is inconsistent or contradicts itself. If contradictions are found, return all sentences containing them; if none are found, respond with "None."

B Supplement to the Experimental Section

B.1 The setting of the S_i hyperparameter.

We conducted a hyperparameter study to select the optimal penalty value for S_i when the generated code is not executable. We compared several candidate values ($S_i = 0.01,\ 0.1,\ 0.2,\ 0.3$), and for each, evaluated (1) the executable code ratio over three rounds of alternating training, and (2) the final consistency metric (hallucination rate) on the CNN/Daily Mail dataset using a Llama-based model. The results are shown in Table 5.

Table 5: Influence of penalty S_i on executable code ratio (Exec. Ratio) over three alternating training rounds and consistency (all values in %).

S_i	Exec. Ratio (1st)	Exec. Ratio (2nd)	Exec. Ratio (3rd)	Consistency
0.01	84.33%	85.77%	85.26%	87.42%
0.1	**87.28%**	**88.50%**	**88.30%**	**91.08%**
0.2	86.40%	87.30%	87.62%	89.24%
0.3	85.02%	85.50%	85.14%	88.01%

As shown in the table, $S_i = 0.1$ achieves the best trade-off in both executable code ratio and consistency. Smaller values (e.g., $S_i = 0.01$) lead to lower and less stable performance, while larger values degrade the results. Thus, we set $S_i = 0.1$ based on optimal empirical outcomes.

B.2 Error Analysis of Low-Quality Code

We analyzed the generated code samples identified as low quality by the quality assessment module. After the third round of alternating training, we designated samples with similarity scores below a threshold of 0.8 as low quality, accounting for 13.97% of the data, while the remaining 86.03% exceeded this threshold. In addition, there were also code samples that remained non-executable after three rounds of alternating training. To gain insight into the nature of these low-quality cases, we conducted a detailed error analysis by manually evaluating a random subset of 100 non-executable samples and 100 samples with similarity scores below 0.8. This analysis revealed four main categories of errors, as illustrated in Fig. 8. These findings highlight the limitations of the current system and suggest promising directions for future research.

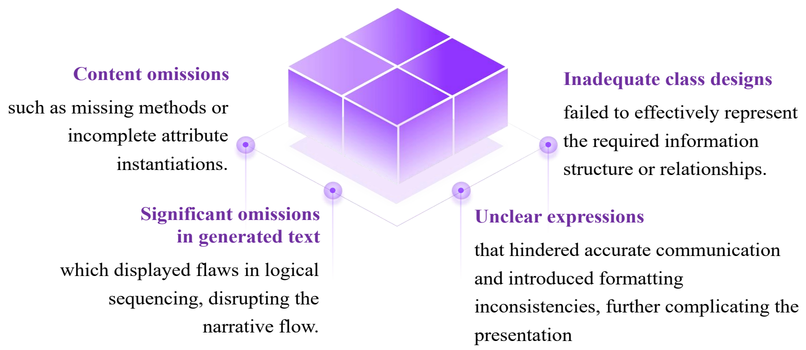

Content omissions
such as missing methods or incomplete attribute instantiations.
Inadequate class designs
failed to effectively represent the required information structure or relationships.
Significant omissions in generated text
which displayed flaws in logical sequencing, disrupting the narrative flow.
Unclear expressions
that hindered accurate communication and introduced formatting inconsistencies, further complicating the presentation

Figure 8: Classification of error causes for generated code that remained low-quality after three rounds of alternating training.

C A Complete Example of Text with Corresponding Generation Code

Original Text: He was born in Santiago de Cuba, Cuba. He is Wilfredo León Hechavarría and Alina Venero Boza (a former volleyball player)'s son. He studied at Escuela Nacional del Voleibol Cubano. León Venero lived in Poland for a while with his Polish girlfriend. On July 14, 2015 he received Polish citizenship and expressed his desire to play in the Poland men's national volleyball team. On June 24, 2016 he married Małgorzata (born Gronkowska). On May 13, 2017 their daughter Natalia was born.

Corresponding Generation Code:

```
# Class representing a place with a name attribute
class Place:
    def __init__(self, name):
        self.name = name
# Class representing an education record (institution attended)
class Education:
    def __init__(self, institution):
        self.institution = institution
# Class representing a period of residence for a person at a specific location
class Residence:
    def __init__(self, person, location):
        self.person = person
        self.location = location
# Class representing citizenship information (person, country, and date obtained)
class Citizenship:
    def __init__(self, person, country, date):
        self.person = person
        self.country = country
        self.date = date
# Class representing a relationship (e.g., parent-child)
class Relationship:
    def __init__(self, person, title):
        self.person = person
        self.title = title
# Class representing a marriage
class Marriage:
    def __init__(self, person, spouse, date):
        self.person = person
        self.spouse = spouse
        self.date = date
# Class representing a child
class Child:
    def __init__(self, person, child, date):
        self.person = person
        self.child = child
        self.date = date
# Class representing a person and relevant personal relationships
class Person:
    def __init__(self, first_name, last_name, title=""):
        self.first_name = first_name
        self.last_name = last_name
        self.title = title
        self.parents = [ ]
        self.spouse = None
        self.children = [ ]
    #Method to add a parent object to the person's parent list
    def add_parent(self, parent):
        self.parents.append(parent)
    #Method to set spouse and marriage date
    def marry(self, spouse, date):
        self.spouse = (spouse, date)
    #Method to add a child with optional date of birth
    def add_child(self, child, date_of_birth=None):
        self.children.append((child, date_of_birth))
# Create instances for Wilfredo León Hechavarría and Alina Venero Boza
```

```
wilfredo_leon_hechavarria = Person(first_name="Wilfredo", last_name="León Hechavarría")
alina_venero_boza = Person(first_name="Alina", last_name="Venero Boza")
# Create an instance for Wilfredo León Hechavarría's father relationship
wilfredo_leon_hechavarria_father = Relationship(person=wilfredo_leon_hechavarria, title="son")
# Create an instance for Wilfredo León Hechavarría's mother relationship
wilfredo_leon_hechavarria_mother = Relationship(person=wilfredo_leon_hechavarria, title="son")
# Create the main person and set parents
leon = Person("Wilfredo", "León Venero")
leon.add_parent(wilfredo_leon_hechavarria)
leon.add_parent(alina_venero_boza)
# Create an education record for the person
escuela = Education("Escuela Nacional del Voleibol Cubano")
escuela_nacional_del_voleibol_cubano = Education(institution="Escuela Nacional del Voleibol Cubano")
# Create a residence record for the person
residence = Residence(leon, "Poland")
wilfredo_leon_hechavarria_residence = Residence(person=wilfredo_leon_hechavarria, location="Poland")
# Create a citizenship record for the person
citizenship = Citizenship(leon, "Poland", "2015-07-14")
wilfredo_leon_hechavarria_citizenship = Citizenship(person=wilfredo_leon_hechavarria, country="Poland",
date="2015-07-14")
# Create a spouse object and connect marriage to the main person
malgorzata = Person("Małgorzata", "Gronkowska")
leon.marry(malgorzata, "2016-06-24")
malgorzata_instance = Person(first_name="Małgorzata", last_name="Gronkowska")
wilfredo_leon_hechavarria_marriage =
Marriage(person=wilfredo_leon_hechavarria, spouse=malgorzata_instance, date="2016-06-24")
# Create a child object and add it to the main person with date of birth
natalia = Person("Natalia", "León Venero")
leon.add_child(natalia, "2017-05-13")
natalia_instance = Person(first_name="Natalia", last_name="León Venero")
wilfredo_leon_hechavarria_child
= Child(person=wilfredo_leon_hechavarria, child=natalia_instance, date="2017-05-13")
```

Reorganized Text:
Note: For better readability, the content is displayed with appropriate line breaks.

He was born in *Place(name='Santiago de Cuba').name*, Cuba.
He is *wilfredo_leon_hechavarria.title wilfredo_leon_hechavarria.first_name wilfredo_leon_hechavarria.last_name* and *alina_venero_boza.title alina_venero_boza.first_name alina_venero_boza.last_name*'s *wilfredo_leon_hechavarria_father.title*.
He studied at *escuela_nacional_del_voleibol_cubano.institution*.
wilfredo_leon_hechavarria.title wilfredo_leon_hechavarria.first_name wilfredo_leon_hechavarria.last_name lived in *wilfredo_leon_hechavarria_residence.location* for a while with his Polish girlfriend.
On *wilfredo_leon_hechavarria_citizenship.date*, he received Polish citizenship
and expressed his desire to play in the Poland men's national volleyball team.
On *wilfredo_leon_hechavarria_marriage.date*, he married *malgorzata_instance.title malgorzata_instance.first_name malgorzata_instance.last_name*.
On *wilfredo_leon_hechavarria_child.date*, their daughter *natalia_instance.title natalia_instance.first_name natalia_instance.last_name* was born.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the Abstract, we clearly elucidate our contributions, and at the end of Section 1 Introduction, we further detail our contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of the work are discussed near the end of the paper, under the "Limitations" heading.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the implementation details in Section 4 to facilitate reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We use publicly available datasets, and we will provide a anonymized link to the code repository.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide details about datasets, inference, evaluation, hyper-parameters, and baseline in Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All reported results in this paper are statistically significant ($p < 0.05$) over five random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discussed the GPU we used to run our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: We consider this work to be primarily technical research that has not yet been applied to specific real-world scenarios. Therefore, we believe it does not directly produce social impacts

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We doesn't release data or models that have a high risk. This paper poses no such risks. .

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the existing assets including code and data used in this paper are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: : The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This research does not involve the use of LLMs in any important, original, or non-standard component of the core methods.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.