When Copying Is Hard: Copy-Constrained Decoding for Exact Span Reproduction

Jinghui Zhang[1] **Lang Gao**[1] **Zongfang Liu**[2,3] **Ruihong Zeng**[1]
Zirui Song[1] **Rui Yan**[4,5] **Kentaro Inui**[1,6,7] **Xiuying Chen**[1*]
[1]MBZUAI [2]Zhejiang University [3]Westlake University
[4]Wuhan University [5]Wuhan Textile University [6]Tohoku University [7]RIKEN
{jinghui.zhang, kentaro.inui, xiuying.chen}@mbzuai.ac.ae

Abstract

Precisely reproducing a target span verbatim within a long context is a fundamental capability test for LLMs: even when the target string is present in context, models often still drift within the span or fail to stop correctly, revealing limitations in source grounding and boundary control. This capability has broad practical relevance. For instance, code generation often requires embedding sensitive strings such as API keys exactly as they appear in context. In this work, we construct a benchmark to test exact span reproduction ability across contracts, scientific literature, code, and random sequences, and find that current models still make surprisingly frequent errors. To address this, we propose **CopyGen**, a lightweight copy-constrained plugin for frozen LLMs that turns copying from a byproduct of next-token prediction into a controlled process. Our key idea is to decompose copying into three distinct decisions: when to copy, how far to copy, and when to stop. Concretely, CopyGen first determines whether to enter copy mode, then predicts a tentative copy length, and finally verifies the generated span token by token. During verification, we adaptively adjust the logits of copy continuation tokens to incentivize or suppress continuation, and emit only the longest valid prefix. Across three LLM backbones and four benchmark tasks, CopyGen improves average exact match by 35% relative without model finetuning, and integrates seamlessly with SFT to further improve it by 4%, while substantially accelerating generation. These results suggest that reliable exact copying is not something LLMs obtain for free, but a decoding-time control problem that requires explicit modeling. Code is available at `https://github.com/mbzuai-nlp/CopyGen`.

1 Introduction

Precisely reproducing long spans of text verbatim, referred to as instruction-conditioned long-span exact copying, can be used to evaluate large language models' ability in exact copying and controlled generation. Although it appears simple, this task places nontrivial demands on models' capabilities in sequence alignment, boundary identification, and fine-grained control over generation. At the same time, this capability has important practical value. In software engineering workflows, models must preserve API keys, identifiers, or configuration strings exactly as they appear in context [11]. In legal processing, they must reproduce clauses verbatim rather than paraphrase them [26]. In such settings, a single incorrect character can break execution or alter the meaning of a legally sensitive passage. Furthermore, when the desired span is already present in the input, regenerating it token by token is both error-prone and computationally unnecessary, making exact copying not only a practically important capability but also an opportunity for improved efficiency.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Despite its practical importance, long-span exact copying has not been studied as a capability in its own right. Existing work on copying falls into two lines, neither of which addresses this problem directly. Architectural copy mechanisms such as CopyNet [8] and pointer-generator networks [25] help models reuse source tokens to improve generation quality, but do not target character-level exactness. Efficiency-oriented methods such as speculative decoding [13], LLMA [28], and CopySpec [6] accelerate inference through multi-token proposal or reuse, with correctness verified against the base model's own predictions rather than strengthened by the copy process itself.

To make this problem concrete, we construct a benchmark for instruction-conditioned exact span reproduction that focuses on preserving the required source text verbatim during generation, rather than broader questions of semantic faithfulness. Specifically, the benchmark covers three real-world tasks, namely legal contracts, scientific literature, and code contexts, where even minor deviations are unacceptable. We further include a synthetic random-sequence task to stress-test copying performance under extreme conditions. Using this benchmark, we evaluate a range of LLMs and find that standard decoding fails consistently across model families and sizes. The errors mainly take two characteristic forms. Internal token mismatches arise when the model drifts mid-span, substituting a plausible-but-wrong token for the correct one. Boundary overruns occur when the model continues generating along the source span even after the target span should have ended. These failures suggest that exact copying requires stronger control than standard next-token prediction provides, calling for an explicit copying mechanism that can maintain source alignment and enforce the correct stopping boundary.

To tackle this issue, we propose **CopyGen**, a lightweight copy-constrained plugin for frozen LLMs. The key idea is to treat exact copying as a controlled decoding problem rather than a byproduct of next-token prediction. CopyGen decomposes exact copying into three coupled decisions: when to copy, how far to tentatively copy, and when to stop. Concretely, at each decoding step, CopyGen measures how strongly the current output prefix is anchored to the source. Once sufficient grounding is detected, the plugin proposes a tentative copy span and replays it token by token against the frozen model. Our analysis reveals tokens that should be copied are predicted with a large logits margin over competing alternatives, while errors, whether mid-span drifts or boundary overruns, concentrate precisely where that margin narrows. During verification, we utilize a learned bias to calibrate the margin at each replayed position, reinforcing source-following where confidence is high and suppressing it where the boundary is near, emitting only the longest prefix that clears the threshold.

Across three LLM backbones and four benchmark tasks, CopyGen delivers substantial improvements in both copy accuracy and decoding efficiency. When applied to frozen models, it improves average exact match by approximately 35% relative. When combined with supervised fine-tuning (SFT), CopyGen continues to provide gains, further increasing average exact match by about 4% relative while also significantly improving decoding throughput. These results suggest that reliable exact copying is not an emergent byproduct of LLM training, but rather a capability that requires explicit control at decoding time along with dedicated copying mechanisms.

In summary, our contributions are threefold: (1) We formulate instruction-conditioned long-span exact copying as a distinct evaluation problem and introduce a dedicated benchmark for studying it. (2) We propose CopyGen, a lightweight copy-constrained plugin for frozen LLMs that augments frozen LLMs with explicit copy control during decoding. (3) We demonstrate that CopyGen substantially improves exact-copy reliability and decoding throughput across model families and scales, and remains effective when combined with supervised finetuning.

2 Related Work

Grounded and Faithful Generation. A closely related line of work studies how to make model outputs stay faithful to an external source rather than drifting toward unsupported continuations. This includes constrained generation frameworks for truthful dialogue grounded in symbolic representations [7], methods for verifiable generation with fine-grained citations to supporting documents [2], and approaches that improve context adherence by adapting the model or prompt at inference time [31]. In retrieval-augmented settings, faithfulness has also been studied as the problem of making generation follow retrieved evidence, especially when retrieved context conflicts with the model's parametric knowledge [29]. These works constrain generation to respect an external source, but target semantic faithfulness rather than character-level exactness. Our setting is stricter: even a single character error or formatting corruption already constitutes failure.

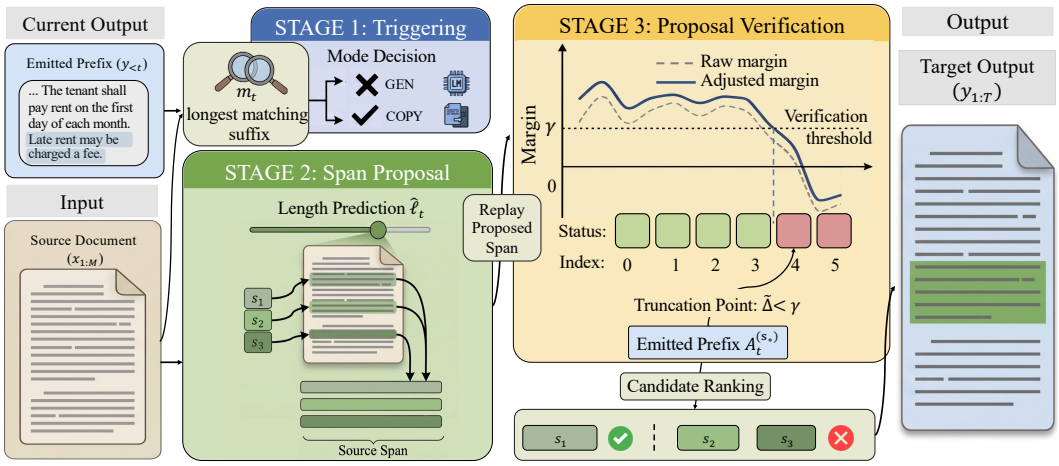

Figure 1: **Overview of CopyGen architecture.** The system operates in three stages: (1) Copy Triggering: determining when to enter copy mode based on grounding length m_t and a learned trigger head; (2) Span Proposal: predicting an adaptive copy length $\hat{\ell}_t$ in log space; and (3) Proposal Verification: calibrating token margins to decide the final emitted prefix $A_t^{(s)}$. This design supports exact copying while maintaining the flexibility of the base LM.

Constrained and Controlled Decoding. Constrained Decoding aims to control model behavior directly at inference time instead of modifying the full model. Early methods were primarily designed to enforce formal validity, for example by restricting generation to syntactically admissible outputs, as in PICARD [24] for text-to-SQL decoding. More recent approaches have extended this idea beyond syntax to broader inference-time control, including symbolic or grammar-based constraints in grounded generation [7, 22], relation-level constraints [3], energy-based constrained generation [23], and efficient non-invasive constraint enforcement [1]. CopyGen is similarly a decoding-time intervention, but differs in the type of constraint it imposes. It is a source-grounded copy constraint that evolves online with the generated prefix, requiring decisions not only on whether to copy, but also how far to extend and where to stop.

Copying and Exact Reproduction in Language Models. Copy mechanism has a long history in sequence generation. Early neural models introduced explicit copy distributions or pointer mechanisms to reproduce source tokens [8, 25], and later extended them to more explicit span-level copying and alignment [21]. More recent inference-time methods also reuse source spans during decoding, primarily to accelerate generation [28, 6] or improve generation quality [14, 15], rather than to ensure exact reproduction itself. A separate thread studies copying as emergent or potentially risky behavior, including controlling memorized content extraction [20], studying probabilistic extraction of memorized content [9], and analyzing copying as a pretraining ability [17]. Our work differs from all of these: we focus on instruction-conditioned exact reproduction of spans explicitly present in the current context, and pair this with a dedicated benchmark and a decoding mechanism designed to improve exact-copy reliability.

3 Method

We consider instruction-conditioned exact copying from long source documents. Each example is a triple $(x_{1:M}, r, y_{1:T})$, where $x_{1:M}$ is a source document, r is a user instruction, and $y_{1:T}$ is the target output, all represented in the token space of a frozen base LM with vocabulary $\mathcal{V}$. The target $y_{1:T}$ consists of two interleaved components: spans copied verbatim from the source, and novel tokens not present in the source. Our goal is to recover $y_{1:T}$ exactly: for copied spans, every character, boundary, and formatting detail must be preserved; for novel tokens, generation should proceed faithfully as usual. Our preliminary experiments show that standard autoregressive decoding is brittle for copy-heavy outputs: small local mistakes can truncate a copied span, omit formatting, or continue beyond the correct boundary. We therefore propose CopyGen, a lightweight copy-constrained method for frozen LLMs that governs three decisions: when to start copying (§3.1), how far to propose copying (§3.2), and how much of the proposal to verify and emit (§3.3).

3.1 Copy Triggering

Reliable copying requires knowing precisely when to start. Entering copy mode too early risks producing spurious repetitions, while entering too late causes the model to miss the correct span entirely. We therefore design a two-stage trigger: first measuring how strongly the current prefix is anchored to the source, then using a learned head to make the final decision.

At decoding step t, let $y_{<t}$ denote the emitted prefix. We define the grounding length

$$m_t = \max\{\ell \in \{0, \dots, \min(t-1, M)\} : \exists j \in \{\ell, \dots, M\},\ x_{j-\ell+1:j} = y_{t-\ell:t-1}\}, \tag{1}$$

where m_t is the length of the longest suffix of the current output prefix that appears contiguously in the source. We also collect the continuation positions compatible with this longest grounded suffix:

$$\mathcal{C}_t = \{j + 1 : x_{j-m_t+1:j} = y_{t-m_t:t-1},\ j < M\}, \tag{2}$$

Each element of $\mathcal{C}_t$ points to a source position that can plausibly continue the current prefix. The scalar m_t provides a compact measure of source grounding, ensuring that copy mode is not activated by accidental overlap on a common token or punctuation mark. If m_t is below a threshold τ, the decoder remains in ordinary generation mode; otherwise the grounding state is passed to CopyGen together with the frozen model hidden state.

Even when the prefix is strongly anchored to the source, the model may already be near the end of the relevant span, or the remaining continuation may no longer belong to the desired target. We therefore introduce a trigger head that jointly considers the hidden state and the grounding signal to make the final decision. Let $h_t \in \mathbb{R}^d$ denote the final hidden state of the frozen base model at decoding step t. We combine it with the grounding signal to form the method input $u_t = \mathrm{LN}([h_t; \log(1 + m_t)])$, where $\mathrm{LN}(\cdot)$ denotes layer normalization. The trigger head then predicts whether to enter COPY or remain in ordinary generation (GEN):

$$p_t^{\mathrm{trig}} = \mathrm{softmax}(W_{\mathrm{trig}} u_t + b_{\mathrm{trig}}). \tag{3}$$

At inference time, copying is activated only when the trigger selects COPY. Otherwise, decoding remains in GEN and falls back to standard one-token generation from the base model.

3.2 Span Proposal

Once copy mode is activated, the model must decide how far to follow the current source continuation. A fixed copy length is too rigid for this purpose. If it is too small, copying must be repeated over many rounds, increasing both overhead and creating more opportunities for transition errors. If it is too large, the proposal is more likely to cross the correct boundary, making verification harder and increasing the risk of over-copy. Recognizing that a fixed copy length is suboptimal, we instead predict a *tentative* copy length adaptively from the current state and leave the final decision to the verifier, which can still truncate the proposal if necessary. To make this prediction stable under the highly skewed distribution of copy lengths, we predict length in log space. Specifically, using the same plugin input u_t defined in §3.1, the length head $f_{\mathrm{len}}(\cdot)$ predicts a tentative copy length:

$$\hat{\ell}_t = \left\lfloor \exp\big(\mathrm{softplus}(f_{\mathrm{len}}(u_t))\big) \right\rfloor. \tag{4}$$

Here $\hat{\ell}_t$ is the proposed number of source tokens to copy from the current aligned continuation. Candidate source starts are drawn from the aligned continuation set $\mathcal{C}_t$, which together with the proposed length $\hat{\ell}_t$ defines the copy span to be verified.

3.3 Proposal Verification

Given a proposed source span, the verifier's role is to determine how much of it can be safely emitted. To inform this decision, we conduct an empirical analysis on output token logits during decoding, as shown in Figure 2. For a candidate source token x_j and the frozen model's next-token logit vector o, we define the ***source margin*** as $\Delta = o[x_j] - \max_{v \in \mathcal{V},\, v \neq x_j} o[v]$, i.e., the gap between the source-token logit and the strongest alternative-token logit. A key observation is that most correctly copied tokens are preferred by a large margin, so these decisions are highly confident and far from the boundary. By contrast, copy errors concentrate near the decision boundary. Internal mismatches typically occur when the correct source token falls slightly *below* the strongest competing token.

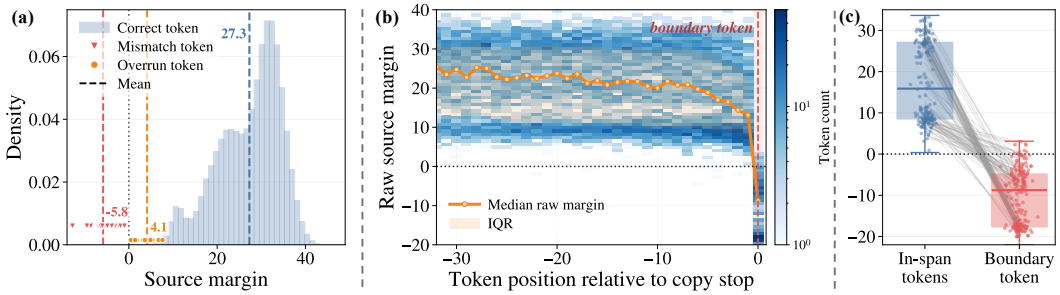

Figure 2: Source-margin behavior for Qwen3-8B. (a) Correct copy tokens typically have large positive source margins, while mismatch and overrun cases lie close to the decision boundary. (b) Around copy-stop positions, the raw source margin stays positive within the copied span and drops sharply at the boundary token. (c) At the span level, in-span tokens tend to have much higher source margins than the boundary token.

Overrun errors, in contrast, often occur when the next source token remains only slightly *above* the strongest alternative, even though generation should have left the copied span. This pattern suggests that the verifier should not replace the base model's preference altogether, but should apply a small adaptive correction precisely in these boundary cases. We implement this through margin calibration, prefix acceptance, and a boundary-focused training objective.

Margin calibration. Since errors concentrate near the decision boundary, we calibrate the raw source margin at each proposal token with a lightweight bias head. For a candidate source start $s \in \mathcal{C}_t$, let $\pi_t^{(s)} = x_{s:s+\hat{\ell}_t-1}$ denote the proposed source span, and let $\pi_{t,k}^{(s)}$ denote its k-th token. We replay this proposal token by token with the frozen base model, obtaining logits $o_{t,k}^{(s)} \in \mathbb{R}^{|\mathcal{V}|}$ at the k-th proposal position, and compute the raw source margin:

$$\Delta_{t,k}^{(s)} = o_{t,k}^{(s)}[\pi_{t,k}^{(s)}] - \max_{v \in \mathcal{V},\, v \neq \pi_{t,k}^{(s)}} o_{t,k}^{(s)}[v]. \tag{5}$$

A bias head $f_{\text{bias}}(\cdot)$ computes a bias term $b_{t,k}^{(s)} = \alpha \tanh\big(f_{\text{bias}}(\text{LN}([\bar{h}_{t,k}^{(s)}; \Delta_{t,k}^{(s)}]))\big)$, where $\bar{h}_{t,k}^{(s)}$ is the replay hidden state at the k-th proposal position and α sets the maximum correction magnitude. We then calibrate the raw margin as $\widetilde{\Delta}_{t,k}^{(s)} = \Delta_{t,k}^{(s)} + b_{t,k}^{(s)}$. Intuitively, the bias head learns when to reinforce a source token and when to suppress it: for internal mismatch cases it can push an under-ranked but correct source token above its competitor, and for overrun cases it can push a marginally favored source token back below the alternative that should be generated instead.

Prefix acceptance. Rather than accepting or rejecting a proposal wholesale, we emit only the longest prefix for which every token passes the margin threshold. We define the accepted length as

$$a_t^{(s)} = \max\Big\{L \in \{0, \ldots, |\pi_t^{(s)}|\} : \widetilde{\Delta}_{t,k}^{(s)} > \gamma,\ \forall k = 1, \ldots, L\Big\},$$

where γ is the verifier threshold, and the accepted prefix is $A_t^{(s)} = \pi_t^{(s)}[1 : a_t^{(s)}]$. The decoder emits the accepted prefix of the best candidate in one step, truncating at the first unsafe token. Candidates are ranked primarily by accepted length, with cumulative adjusted margin and cumulative log-probability as tie-breakers. If no proposal is accepted, the decoder falls back to a one-step source-aware generation using the same bias head; details are provided in Appendix F.

Training objective. As observed in Figure 2, since the vast majority of correct copied tokens tend to have large positive source margins, we train the bias head with a hardness-weighted margin objective that focuses learning on ambiguous boundary cases. For each replayed position i, let $y_i \in \{+1, -1\}$ indicate whether it is an in-span copy token or a boundary-crossing token, and let Δ_i^{raw} and Δ_i^{adj} denote its raw and adjusted margins. We define a hardness weight as:

$$w_i^{\text{hard}} = \sigma\Big(-\tfrac{y_i \Delta_i^{\text{raw}}}{\tau_w}\Big), \tag{6}$$

where σ denotes the logistic sigmoid and τ_w is a temperature. The main verifier objective is

$$\mathcal{L}_{\text{margin}} = \frac{\sum_{i:y_i=+1} w_i^{\text{hard}} [m_+ - \Delta_i^{\text{adj}}]_+}{\sum_{i:y_i=+1} w_i^{\text{hard}} + \varepsilon} + \lambda_- \frac{\sum_{i:y_i=-1} w_i^{\text{hard}} [\Delta_i^{\text{adj}} + m_-]_+}{\sum_{i:y_i=-1} w_i^{\text{hard}} + \varepsilon}, \tag{7}$$

where m_+ and m_- are positive and negative safety margins, $[z]_+ = \max(z, 0)$, λ_- weights the negative term, and ε is a small constant for stability. This encourages in-span copy tokens to remain above the decision boundary and boundary-crossing tokens to remain below it.

Overall, the verifier turns copy acceptance into a calibrated decision process: high-confidence source tokens pass through unchanged, difficult in-span tokens can be rescued by a positive correction, and marginal boundary-crossing tokens can be suppressed before they trigger over-copy. The full decoding procedure is summarized in Appendix C.

4 Experiments

4.1 Experimental Setup

Benchmark Details. We evaluate exact span reproduction on three real-world domains: *Contract*, *Literature*, and *Code*. Across all three domains, the target spans are highly sensitive to boundaries, formatting, and character-level corruption, so even small deviations can make the output unusable.

Contract studies *structured legal clause extraction*, where the model must identify the requested clause(s) from contracts curated from SEC EDGAR filings and place them into a JSON-style structured output format. *Literature* asks for *scientific evidence extraction*, requiring the model to return verbatim supporting spans from full-text articles collected from PubMed Central, including experimental settings, subject or sample descriptions, and statistical analysis statements. *Code* focuses on *code explanation*, where the model must briefly explain a target snippet and then reproduce it exactly, using contexts drawn from widely used open-source Python repositories. To further evaluate exact-copy reliability under stress, we additionally include *Random Sequence*, where the model must precisely copy synthetic OpenSSH-style key sequences from the input, reflecting cases where encoded configuration values or credential-like identifiers must be copied exactly. Full benchmark construction details, data sources, and dataset statistics are provided in Appendix E.

Evaluation Metrics. We evaluate both correctness and efficiency. For correctness, we use Exact Match (EM), which requires the generated output to match the target span exactly. For efficiency, we report decoding throughput in tokens per second (Tokens/s), and summarize relative speedup.

Models. We evaluate on three representative LLMs from two model families and scales: Qwen3-4B, Qwen3-8B [27], and Ministral-8B-Instruct (Ministral-8B hereafter) [18]. This setup tests whether exact-copy behavior and the effect of our method are consistent across model families and sizes.

Baselines. For each base model, we report four configurations following the structure of Table 1: (1) **Base Model**, i.e., the original frozen LLM with standard decoding; (2) **CopyGen**, which applies our decoding-time plugin to the frozen base model; (3) **Base Model$_{\text{SFT}}$**, which applies supervised finetuning using LoRA [10]; and (4) **CopyGen$_{\text{SFT}}$**, which combines our plugin with the SFT-adapted model. This setup allows us to separate the effects of decoding-time control from those of parameter-efficient finetuning, and to test whether the two are complementary.

To examine both the accuracy and efficiency of CopyGen, we further compare it against three families of baselines that exploit the source text during decoding, as reported in Table 2: (1) **Lossless Source-Reuse Acceleration**, including LLMA [28] and CopySpec [6], which speeds up decoding while preserving the base model's output distribution; (2) **Rule-Based Source Constraints**, including a PICARD-style [24] decoder that masks generation to source-consistent continuations and a non-learned Longest-Copy rule that directly copies the longest matched source continuation; and and (3) **Learned Copy Mechanisms**, represented by CopySpan [21], a span-level pointer-generator network trained on top of the frozen LM. We also evaluate three frontier LLMs, GPT-4.1-nano [19], Gemini-2.5-Flash-Lite [4], and DeepSeek-V4-Pro [5], under the same protocol (Table 16).

Implementation Details. For SFT baselines, we use LoRA with rank 16. For CopyGen, we freeze the backbone LM and train only the lightweight plugin (CopyGen$_{\text{SFT}}$ applies the plugin on top of the SFT backbone). For all trainable components, we use dropout 0.05 and gradient accumulation 16. Optimization uses AdamW [16] with a cosine learning-rate schedule. All experiments are run in BF16 precision on one NVIDIA RTX PRO 6000 GPU. To ensure reproducibility and make decoding behavior comparable across methods, we use deterministic decoding with temperature set to 0 in all experiments; for CopyGen, proposal verification uses the positive-margin threshold $\gamma = 0$. Detailed hyperparameters and prompt templates are provided in Appendices G and J, respectively.

Table 1: Overall performance comparison. Average column shows the mean results across all four datasets. For all metrics, higher scores indicate better performance. Better results within each corresponding comparison pair are highlighted in **bold** (two-tailed paired t-test, $p < 0.05$).

Method	Contract		Literature		Code		Random Sequence		Average	
	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s
Qwen3-4B										
Base Model	5.67	66.54	68.28	60.09	54.15	54.75	53.60	62.28	45.43	60.92
CopyGen	**13.48**	**238.02**	**84.83**	**729.42**	**55.02**	**132.28**	**74.80**	**2939.15**	**57.03**	**1009.72**
Base Model$_{\text{SFT}}$	90.07	43.62	84.83	44.35	97.38	43.88	56.00	44.77	82.07	44.16
CopyGen$_{\text{SFT}}$	**90.78**	**160.12**	**85.52**	**149.24**	**98.69**	**424.87**	**90.40**	**2143.36**	**91.35**	**719.40**
Qwen3-8B										
Base Model	16.31	53.00	77.24	56.30	80.35	56.39	67.20	56.18	60.28	55.47
CopyGen	**34.75**	**219.54**	**86.21**	**592.85**	**96.51**	**72.79**	**90.40**	**2483.23**	**76.97**	**842.10**
Base Model$_{\text{SFT}}$	**96.45**	43.21	**94.48**	43.92	97.82	42.52	90.00	43.62	94.69	43.32
CopyGen$_{\text{SFT}}$	**96.45**	**159.33**	**94.48**	**234.16**	**98.25**	**326.87**	**96.80**	**1333.66**	**96.50**	**513.51**
Ministral-8B										
Base Model	19.15	60.01	41.38	62.05	96.07	63.44	42.80	60.93	49.85	61.61
CopyGen	**22.70**	**204.22**	**88.28**	**521.48**	**96.51**	**69.11**	**100.00**	**2609.59**	**76.87**	**851.10**
Base Model$_{\text{SFT}}$	**98.58**	49.42	**92.41**	49.43	99.56	49.48	98.00	49.97	97.14	49.58
CopyGen$_{\text{SFT}}$	**98.58**	**178.88**	**92.41**	**529.07**	**100.00**	**400.70**	**100.00**	**1939.16**	**97.75**	**761.95**

Table 2: Comparison with source-reuse and constrained-decoding baselines. Results are averaged over the three backbones (Qwen3-4B, Qwen3-8B, Ministral-8B); per-backbone results are in Table 15. Throughput of Longest-Copy is omitted (–) as it is not meaningful without model verification. Best results in each column are highlighted in **bold** (two-tailed paired t-test, $p < 0.05$).

Method	Contract		Literature		Code		Random Sequence		Average	
	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s
Base Model	13.71	59.85	62.30	59.48	76.86	58.19	54.53	59.80	51.85	59.33
Lossless Source-Reuse Acceleration										
LLMA	13.71	**238.60**	62.30	493.40	76.86	109.70	54.53	288.27	51.85	282.49
CopySpec	13.71	183.80	62.30	301.23	76.86	111.77	54.53	171.77	51.85	192.14
Rule-Based Copy Constraint										
PICARD-style	0.00	54.27	62.53	54.73	89.37	52.73	54.53	180.17	51.61	85.48
Longest-Copy	0.00	–	12.41	–	**92.87**	–	0.00	–	26.32	–
Learned Copy Mechanism										
CopySpan	3.31	46.78	19.54	419.88	85.01	**336.48**	20.40	1422.80	32.06	556.48
CopyGen (Ours)	**23.64**	220.59	**86.44**	**614.58**	82.68	91.39	**88.40**	**2677.32**	**70.29**	**900.97**

4.2 Main Results

Table 1 reports exact-match accuracy and decoding throughput across four benchmark domains and three backbone LLMs, and Figure 3 visualizes throughput gains on the three real-world tasks. *(1) CopyGen substantially improves exact-copy fidelity and decoding efficiency for frozen LLMs.* Averaged over the three frozen backbones, CopyGen raises EM from 51.9 to 70.3 while increasing throughput from 59.3 to 901.0 tokens/s, corresponding to roughly a $15.2\times$ speedup. The gains are especially strong on *Literature* and *Random Sequence*, where long-span exact reproduction is most brittle under standard decoding. For example, the average EM improves from 60.3 to 77.0 on Qwen3-8B and from 49.9 to 76.9 on Ministral-8B, showing that CopyGen strengthens exact copying without updating base model parameters. *(2) CopyGen remains highly effective when combined with supervised finetuning.* With SFT, CopyGen further improves the average EM from 91.3 to 95.2, while increasing throughput from 45.7 to 665.0 tokens/s, a $14.6\times$ speedup. These results show that decoding-time copy control and parameter-efficient adaptation are complementary: finetuning improves the model's overall task performance, while CopyGen further strengthens exact reproduction and substantially accelerates generation. As shown in Figure 3, these speedups are consistent across real-world domains, reaching over $12\times$ in the base model setting and over $10\times$ in the SFT setting.

(3) The largest gains appear precisely on tasks where exact boundary and character-level fidelity matter most. On *Contract*, *Literature*, and *Random Sequence*, even small local mistakes in punctuation, clause boundaries, or long low-redundancy strings invalidate the output, and CopyGen yields the largest EM improvements. On *Code*, the EM gains are more moderate but still consistent, while throughput improves substantially across all models. This pattern indicates that CopyGen is effective at stabilizing long source-grounded spans and turning many fragile token-level decisions into fewer verified span emissions. *(4) CopyGen is the only method that improves both accuracy and efficiency over source-reuse and constrained-decoding baselines.* As shown in Table 2, lossless acceleration methods such as LLMA and CopySpec leave EM at the base-model level, whereas CopyGen improves average EM by 18.44 points while running $3.19\times$ and $4.69\times$ faster than them. Rule-based constraints and CopySpan all fall below the base model on average, since they cannot reliably determine where the requested span should start and stop in the source document, and the PICARD-style decoder even fails entirely on *Contract*, where copied clauses must be embedded in JSON strings. *(5) Exact copying remains challenging even for frontier LLMs.* Even the strongest frontier LLM, DeepSeek-V4-Pro, reaches only 56.03 EM on *Contract* (Table 16). With CopyGen, Qwen3-8B and Ministral-8B already surpass GPT-4.1-nano and Gemini-2.5-Flash-Lite without finetuning, and all three backbones outperform every frontier LLM when further combined with SFT.

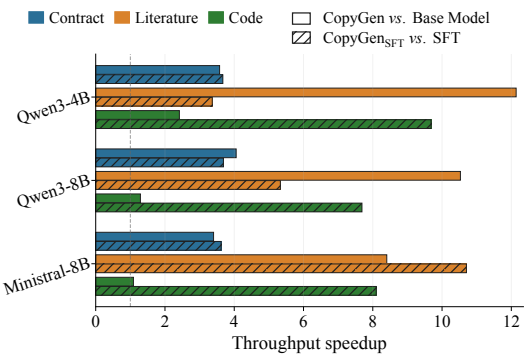

Figure 3: Throughput speedup on real-world tasks.

5 Analysis and Discussion

5.1 Ablation Study

We also conduct an ablation study to examine the contribution of each key component in CopyGen, as shown in Tables 3 and 17. For *w/o Length Proposal*, we replace the predicted tentative span length with a fixed copy length $k \in \{8, 16, 32, 64, 128, 256\}$ and average results across the sweep. The throughput drop is the most pronounced, confirming that no single fixed value can consistently balance boundary precision and long-span efficiency across tasks. Shorter fixed lengths tend to be conservative and avoid over-copy, but they require more copy rounds and limit throughput gains, while longer ones improve speed at the cost of crossing copy boundaries more frequently. When all replayed positions are assigned equal training weight (*w/o Adaptive Weighting*), both exact match and throughput fall clearly, as learning becomes dominated by easy in-span copy tokens rather than the ambiguous near-boundary cases where copy failures actually occur. The steepest degradation comes from removing margin calibration (*w/o Margin Calibration*), which effectively reduces CopyGen to copying only when the base model already favors the source token. In this case, the verifier can neither rescue under-ranked in-span tokens nor suppress marginal continuations near the true boundary, leaving the most error-prone decoding decisions unguarded and entirely to the frozen base model. Overall, length proposal drives efficiency, adaptive weighting focuses learning where it matters, and margin calibration is what ultimately turns that supervision into reliable exact-copy control.

Table 3: Average ablation results of CopyGen across the four benchmark domains. Best results are shown in **bold**.

Model	Variant	EM (↑)	Tokens/s (↑)
Qwen3-4B	**Full**	**57.03**	**1009.72**
	w/o Length Proposal	56.69	452.49
	w/o Adaptive Weighting	46.78	690.55
	w/o Margin Calibration	44.94	798.14
Qwen3-8B	**Full**	**76.97**	**842.10**
	w/o Length Proposal	74.32	354.94
	w/o Adaptive Weighting	63.67	656.22
	w/o Margin Calibration	60.47	670.12
Ministral-8B	**Full**	**76.87**	**851.10**
	w/o Length Proposal	74.95	298.94
	w/o Adaptive Weighting	60.89	292.49
	w/o Margin Calibration	50.03	243.38

5.2 Copy Dynamics and Calibration

We analyze the copy behavior of CopyGen to understand whether it simply copies longer spans or learns to control when copying should stop. Using the literature domain as an example, Figure 4(a) shows that CopyGen follows a propose-then-verify pattern: the median proposed length is 197

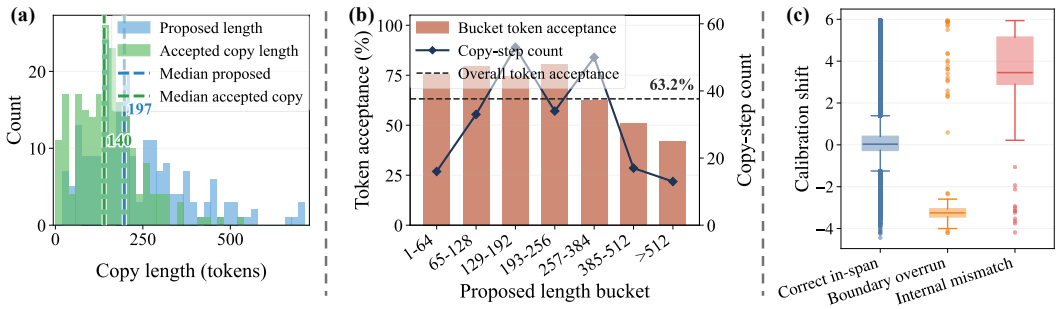

Figure 4: Copy proposal dynamics and calibration behavior. (a) Proposed copy spans are often longer than the spans finally accepted by the verifier. (b) Token acceptance remains high for medium-length proposals but drops for very long proposals, indicating conservative truncation near uncertain boundaries. (c) Calibration bias suppresses boundary overruns while boosting internally fragile copy positions, helping preserve exact in-span reproduction. Results shown for Qwen3-8B.

tokens, whereas the median accepted length is 140 tokens. This gap shows that CopyGen often proposes broad source continuations and lets the verifier trim them before emission. The results grouped by proposal length further reveal where this strategy is most useful. As shown in Figure 4(b), copy-step count rises from short to medium-length proposals and then drops for very long proposals, suggesting that most effective copy events occur at moderate span lengths. At the same time, token acceptance remains high for medium-length proposals but becomes more conservative for very long ones, indicating that verification mainly intervenes when boundary uncertainty increases. The same proposal-and-acceptance trends are observed across other tasks and backbones, with full results provided in Appendix H.4. Finally, the calibration analysis in Figure 4(c) provides token-level evidence for this control mechanism. Confident in-span tokens receive little shift, preserving the base model's decisions. Boundary overrun tokens are pushed downward to prevent over-copying, while internally fragile positions receive positive bias to keep generation aligned with the source.

This analysis suggests that CopyGen improves exact copying not by blindly copying more, but by combining broad span proposals with boundary-aware verification and local calibration.

5.3 Integration with Inference Engines

In real-world deployment, LLMs are typically served through optimized inference engines, and the practical value of CopyGen ultimately depends on whether its gains carry over to such systems. We therefore integrate CopyGen with Qwen3-8B into vLLM [12] and SGLang [30]. As shown in Table 4, CopyGen improves EM by about 27% relative in both engines, matching its gain under standard decoding, and raises throughput to over 2,000 tokens/s, corresponding to average speedups of $5.74\times$ and $5.78\times$. These speedups are obtained on top of engines that already apply batching and highly optimized execution, while the plugin heads add less than 2.7% to wall-clock time, indicating that CopyGen can be readily and seamlessly integrated into existing serving stacks.

Table 4: CopyGen on Qwen3-8B under standard decoding and inference engines, averaged over four domains (full results in Table 13). Relative EM changes are computed against the base model; speedup is the mean of per-domain speedups.

Setting	EM (↑)		Tokens/s (↑)		Speedup
	Base	CopyGen	Base	CopyGen	
Standard	60.28	**76.97** (+27.7%)	55.47	**842.10**	15.04×
vLLM	60.69	**77.00** (+26.9%)	374.57	**2342.67**	5.74×
SGLang	60.67	**77.06** (+27.0%)	366.47	**2074.49**	5.78×

5.4 Training Efficiency Analysis

Beyond improving decoding efficiency at inference time, CopyGen is also lightweight to train. Table 5 summarizes the training resources required by SFT and CopyGen across the three backbone models, with full statistics in Table 18. Since CopyGen freezes the base LM and trains only small plugin heads, it substantially reduces peak GPU memory relative to LoRA SFT across all backbones, making copy-specific adaptation practical even under tight hardware budgets. Despite the lower memory usage, training time per example remains broadly comparable to SFT, with marginal improvements in

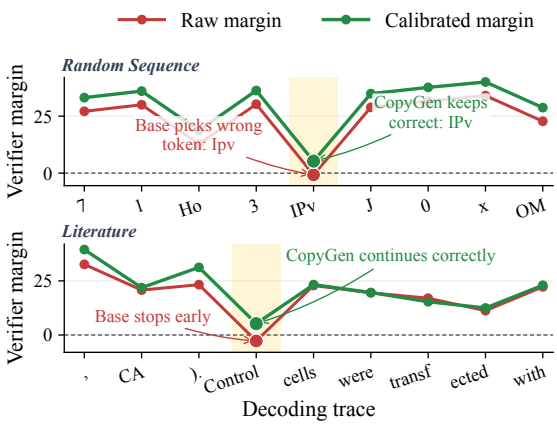

Figure 5: Calibrated decoding trace case study.

Table 5: Average training resource comparison across four tasks. Relative changes are computed against SFT.

Method	Peak GiB (↓)	Train s/item (↓)
	Qwen3-4B	
SFT	27.25	1.935
CopyGen	**10.95** (-59.8%)	**1.608** (-16.9%)
CopyGen$_{\text{SFT}}$	11.00 (-59.6%)	1.724 (-10.9%)
	Qwen3-8B	
SFT	35.55	1.597
CopyGen	**19.01** (-46.5%)	1.600 (+0.2%)
CopyGen$_{\text{SFT}}$	19.06 (-46.4%)	**1.590** (-0.5%)
	Ministral-8B	
SFT	32.51	1.639
CopyGen	**18.65** (-42.6%)	1.649 (+0.6%)
CopyGen$_{\text{SFT}}$	18.68 (-42.5%)	**1.600** (-2.4%)

several settings. Notably, these efficiency and accuracy gains come without touching the base model parameters, making CopyGen a low-overhead, plug-and-play addition to existing LLMs.

5.5 Case Study

To intuitively understand how CopyGen intervenes at the token level, we randomly select two concrete decoding failures and trace what calibration does to fix them. In the first example, the model is copying a random key sequence and stumbles on a single character. The base model produces `...Gcw71Ho3Ipv J0xOMk...`, swapping the capital `P` for a lowercase `p` mid-span. This is not a boundary error: the model is well inside an active copy region, and every surrounding token is decoded correctly. The top panel of Figure 5 shows why it happens: the raw verifier margin for `IPv` dips just below zero at that position, and the base model tips toward the wrong token. Calibration pushes the margin back above the threshold, and CopyGen emits the correct `...Gcw71Ho3IPv J0xOMk...` without interruption. The second example is a different kind of failure. Asked to extract the full transfection procedure from a scientific article, the base model stops prematurely after the first sentence, treating the period in "...manufacturer instructions (Invitrogen, Carlsbad, CA)." as a natural endpoint. The bottom panel shows the raw margin turning negative at `Control`, the token that should begin the next sentence. The base model reads this dip as a stopping cue, but CopyGen recognizes it as a fragile transition rather than a true boundary and lifts the margin above zero, allowing decoding to continue into "Control cells were transfected with empty vector."

The two cases capture opposite failure modes, one where the model loses track mid-span, and one where it gives up too early. Calibrated verification addresses both by learning to distinguish boundaries from low-confidence positions, rather than deferring to the base model's preference.

6 Conclusion

In this work, we study exact long-span reproduction as a distinct and practically important capability of large language models. Even when the target span is explicitly present in the input, standard decoding often fails to reproduce it faithfully, especially inside copied spans and near their boundaries. We introduce a benchmark for exact copying to make this problem concrete, and find current LLMs fall short in reproducing text exactly. To address this problem, we propose CopyGen, a copy-constrained plugin for frozen LLMs that decomposes exact copying into triggering, tentative span proposal, and token-level verification. By calibrating source-continuation tokens during verification, experimental results show CopyGen improves both exact-match accuracy and decoding efficiency across models and tasks, suggesting that reliable exact copying is not a trivial consequence of next-token prediction, but a decoding-time control problem that deserves explicit modeling. We hope this work encourages further research on reliable text generation with LLMs.

References

[1] L. Beurer-Kellner, M. Fischer, and M. Vechev. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *International Conference on Machine Learning*, 2024.

[2] S. Cao and L. Wang. Verifiable generation with subsentence-level fine-grained citations. In *Findings of the Association for Computational Linguistics*, 2024.

[3] X. Chen, Z. Yang, and X. Wan. Relation-constrained decoding for text generation. In *Advances in Neural Information Processing Systems*, 2022.

[4] G. Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[5] DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

[6] R.-G. Dumitru, M. Yang, V. Yadav, and M. Surdeanu. CopySpec: Accelerating LLMs with speculative copy-and-paste. In *Conference on Empirical Methods in Natural Language Processing*, 2025.

[7] H. Fang, A. Balakrishnan, H. Jhamtani, J. Bufe, J. Crawford, J. Krishnamurthy, A. Pauls, J. Eisner, J. Andreas, and D. Klein. The whole truth and nothing but the truth: Faithful and controllable dialogue response generation with dataflow transduction and constrained decoding. In *Findings of the Association for Computational Linguistics*, 2023.

[8] J. Gu, Z. Lu, H. Li, and V. O. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Annual Meeting of the Association for Computational Linguistics*, 2016.

[9] J. Hayes, M. Swanberg, H. Chaudhari, I. Yona, I. Shumailov, M. Nasr, C. A. Choquette-Choo, K. Lee, and A. F. Cooper. Measuring memorization in language models via probabilistic extraction. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[11] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

[12] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles*, 2023.

[13] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2023.

[14] M. Li, X. Chen, A. Holtzman, B. Chen, J. Lin, W.-t. Yih, and X. V. Lin. Nearest neighbor speculative decoding for LLM generation and attribution. In *Advances in Neural Information Processing Systems*, 2024.

[15] Y. Liu, T. Ji, C. Sun, Y. Wu, and X. Wang. Generation with dynamic vocabulary. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

[16] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[17] A. Lv, R. Xie, X. Sun, Z. Kang, and R. Yan. Language models "grok" to copy. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics*, 2025.

[18] Mistral AI. Model card for Ministral-8B-Instruct-2410. `https://huggingface.co/mistralai/Ministral-8B-Instruct-2410`, 2024.

[19] OpenAI. Introducing GPT-4.1 in the API. `https://openai.com/index/gpt-4-1/`, 2025.

[20] M. Ozdayi, C. Peris, J. FitzGerald, C. Dupuy, J. Majmudar, H. Khan, R. Parikh, and R. Gupta. Controlling the extraction of memorized data from large language models via prompt-tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[21] S. Panthaplackel, M. Allamanis, and M. Brockschmidt. Copy That! editing sequences by copying spans. In *AAAI Conference on Artificial Intelligence*, 2021.

[22] K. Park, J. Wang, T. Berg-Kirkpatrick, N. Polikarpova, and L. D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems*, 2024.

[23] L. Qin, S. Welleck, D. Khashabi, and Y. Choi. COLD decoding: Energy-based constrained text generation with Langevin dynamics. In *Advances in Neural Information Processing Systems*, 2022.

[24] T. Scholak, N. Schucher, and D. Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[25] A. See, P. J. Liu, and C. D. Manning. Get to the point: Summarization with pointer-generator networks. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

[26] S. H. Wang, M. Zubkov, K. Fan, S. Harrell, Y. Sun, W. Chen, A. Plesner, and R. Wattenhofer. ACORD: An expert-annotated retrieval dataset for legal contract drafting. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

[27] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[28] N. Yang, T. Ge, L. Wang, B. Jiao, D. Jiang, L. Yang, R. Majumder, and F. Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023.

[29] Q. Zhang, Z. Xiang, Y. Xiao, L. Wang, J. Li, X. Wang, and J. Su. FaithfulRAG: Fact-level conflict modeling for context-faithful retrieval-augmented generation. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

[30] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. Barrett, and Y. Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

[31] J. Zhu, S. Liu, Y. Yu, B. Tang, Y. Yan, Z. Li, F. Xiong, T. Xu, and M. B. Blaschko. FastMem: Fast memorization of prompt improves context awareness of large language models. In *Findings of the Association for Computational Linguistics*, 2024.

A Limitations

This study has several limitations.

First, our benchmark focuses on exact span reproduction, and therefore primarily captures boundary fidelity and character-level correctness rather than broader forms of faithful generation such as paraphrasing or abstraction. We believe that treating character-level exact copying as a standalone evaluation dimension is worthwhile in its own right. In many practical applications, semantic approximation is simply not enough, and this capability has long lacked dedicated attention in the literature, making our work a meaningful step toward filling that gap.

Second, although we include realistic domains such as contracts, scientific literature, and code, the current benchmark still covers only a limited range of exact-copy scenarios and does not consider settings such as noisy documents, OCR errors, or multilingual copying. We hope the three domains we selected serve as a representative starting point that can inform broader benchmark extensions in future work, rather than an attempt to exhaustively cover every possible scenario.

Third, our method is a decoding-time plugin for frozen LLMs, and its performance remains to some extent dependent on the underlying base model. This design trade-off is intentional. We aimed for CopyGen to integrate with existing models with minimal intrusiveness, and our experiments show that it consistently delivers meaningful gains even when the base model is already quite capable.

Finally, the efficiency gains of our method depend on both the task and the implementation. When copy proposals are short or frequently rejected, the overhead of verification may reduce the practical speed advantage. In addition, our current engine integration operates outside fused CUDA kernels, so on-the-fly suffix grounding introduces branching overhead in the decoding loop. Although this overhead is already reflected in our reported measurements, kernel-level batched suffix matching could reduce it further and is left for future work. That said, long-span, low-redundancy exact copying is a common and particularly challenging setting in practice, and it is also where CopyGen's efficiency gains are most pronounced. This suggests a natural alignment between where our method is designed to help and where it delivers the most value.

B Broader Impact

Reliable exact span reproduction can have positive impacts in applications where faithful reuse of source text is important. For example, it can help code assistants preserve identifiers, configuration values, and code snippets exactly. It can also support legal and scientific workflows that require verbatim evidence extraction, while reducing errors introduced by paraphrasing or token-level regeneration. Since CopyGen can emit verified copied spans more efficiently, it may also reduce inference cost for copy-heavy generation tasks.

At the same time, improving exact copying can introduce risks if used without appropriate safeguards. A system with stronger copying ability may reproduce sensitive strings, private identifiers, copyrighted text, or other protected content when such content appears in the input context. It may also make users overconfident that copied outputs are always appropriate to reuse, even when the source material is confidential, legally restricted, or incorrectly selected. These risks are especially relevant in settings involving credentials, proprietary documents, legal text, or scientific claims.

Our work is intended for source-grounded copying where the target content is explicitly provided in the input and the user or application has permission to use it. We do not train models to extract memorized content from parameters, and the random-sequence benchmark uses only synthetic credential-like strings rather than real secrets. In practical deployments, exact-copy mechanisms should be combined with access control, data-permission checks, and safeguards against copying sensitive or restricted content.

C Algorithm Framework of CopyGen

Algorithm 1 shows the detailed procedure of how CopyGen works. At each step, the decoder first measures whether the current prefix is sufficiently grounded in the source. If so, it may enter copy mode, propose a tentative source span, and verify it token by token. Only the longest verified prefix

Algorithm 1 Copy-Constrained Decoding of CopyGen

Input: source $x_{1:M}$, instruction r, frozen LM f_θ, plugin heads, thresholds τ, γ
Output: generated output y
Build prompt $\mathcal{P}(x_{1:M}, r)$ and initialize $y \leftarrow \emptyset$
while termination not reached **do**
    Compute hidden state h_t and next-token logits o_t
    if $\arg\max o_t = \text{EOS}$ **then**
        break
    end if
    Compute grounding signal $(m_t, \mathcal{C}_t)$ ▷ longest grounded suffix and aligned continuations
    if $m_t < \tau$ **then**
        Emit $\arg\max o_t$ and continue
    end if
    Form $u_t = \text{LN}([h_t; \log(1 + m_t)])$ and predict trigger p_t^{trig}
    if $\arg\max p_t^{\text{trig}} = \text{GEN}$ **then**
        Emit $\arg\max o_t$ and continue
    end if
    Predict tentative copy length $\hat{\ell}_t$ and construct candidate starts $\mathcal{S}_t \subseteq \mathcal{C}_t$
    for all $s \in \mathcal{S}_t$ **do**
        Form proposal $\pi_t^{(s)} = x_{s:s+\hat{\ell}_t-1}$
        Replay $\pi_t^{(s)}$ and compute calibrated margins $\widetilde{\Delta}_{t,k}^{(s)}$
        Compute accepted length $a_t^{(s)} = \max\{L : \widetilde{\Delta}_{t,k}^{(s)} > \gamma, \forall k = 1, \ldots, L\}$
        Set accepted prefix $A_t^{(s)} = \pi_t^{(s)}[1 : a_t^{(s)}]$
    end for
    Select the best candidate $s^\star$
    if $a_t^{(s^\star)} > 0$ **then**
        Emit $A_t^{(s^\star)}$ ▷ longest verified prefix
    else
        Emit one source-aware fallback token
    end if
end while
return y

is emitted. If no proposal token is accepted, the decoder performs a source-aware fallback generation step by reranking candidate source-follow tokens before emitting one token.

D Further Analyses

D.1 Fixed Copy Length

To isolate the effect of adaptive span proposal, we compare CopyGen with variants that replace the predicted tentative span length by a fixed global copy length k. We sweep $k \in \{8, 16, 32, 64, 128, 256\}$ while keeping the remaining decoding and verification procedure unchanged. Figure 6 shows two complementary views. The left panel plots the speed–accuracy tradeoff for each task. Across tasks, CopyGen is consistently close to the best fixed-length tradeoffs: it preserves the accuracy of shorter fixed lengths while achieving throughput closer to longer fixed lengths. The right panel shows the fixed-length sweep directly as a function of k. The results show that no single fixed length is consistently best across tasks and models. Smaller values of k are conservative and help avoid over-copy, but they require more copy rounds and provide limited throughput gains. Larger values of k increase throughput, but they are more likely to cross the correct copy boundary and can reduce exact match. CopyGen avoids this global tuning problem by predicting a tentative length from the current decoding state and leaving the final boundary decision to verification. This adaptive proposal lets CopyGen achieve a stronger speed–accuracy tradeoff than choosing a single fixed copy length.

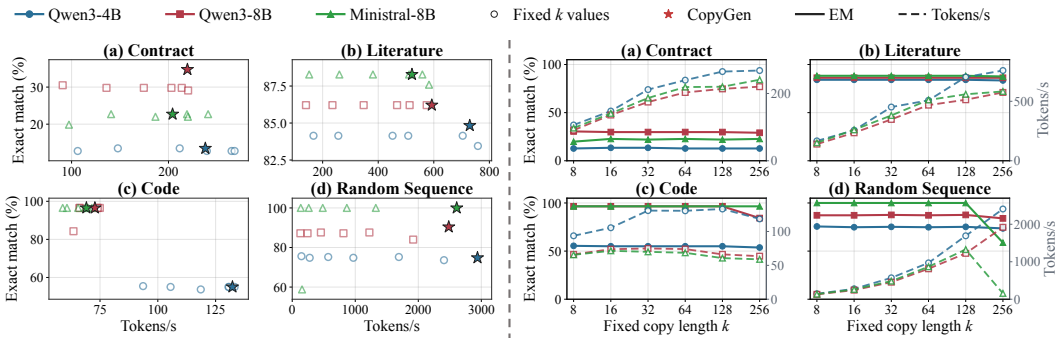

Figure 6: Fixed copy length analysis. Left: speed–accuracy tradeoff between fixed-length copying and CopyGen's adaptive span proposal, shown separately for each task. CopyGen is close to the best fixed-length tradeoffs, preserving the accuracy of shorter fixed spans while approaching the throughput of longer fixed spans. Right: fixed-length sweep over k, where solid lines denote exact match and dashed lines denote throughput. Small fixed lengths preserve accuracy but limit throughput, while large fixed lengths improve throughput but can cross copy boundaries and hurt accuracy.

D.2 Domain Generalization

We further evaluate whether CopyGen generalizes across real-world domains using a leave-one-domain-out setup on Qwen3-8B and Ministral-8B. For each target domain, CopyGen is trained on the other two real-world domains and evaluated directly on the held-out one. As shown in Table 6, the held-out-domain variant performs better than the base model overall, while trailing the corresponding in-domain CopyGen result. This indicates that CopyGen has nontrivial cross-domain generalization ability: the learned verifier transfers to unseen real-world domains and still delivers clear gains over the base model, although in-domain data remains helpful for best performance.

Table 6: Cross-domain generalization of CopyGen under leave-one-domain-out training on the three real-world domains. CopyGen (Held-out) trains on the two non-target real-world domains and evaluates on the held-out domain. Average is computed over Contract, Literature, and Code. Best results are highlighted in **bold**.

Test Domain	Train Domains	Base Model		CopyGen		CopyGen (Held-out)	
		EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)
				Qwen3-8B			
Contract	Literature + Code	16.31	53.00	**34.75**	**219.54**	16.31	162.05
Literature	Contract + Code	77.24	56.30	**86.21**	**592.85**	85.52	391.49
Code	Contract + Literature	80.35	56.39	**96.51**	**72.79**	94.76	68.01
Average	–	57.97	55.23	**72.49**	**295.06**	65.53	207.18
				Ministral-8B			
Contract	Literature + Code	19.15	60.01	**22.70**	**204.22**	21.28	179.81
Literature	Contract + Code	41.38	62.05	**88.28**	**521.48**	68.97	393.91
Code	Contract + Literature	96.07	63.44	**96.51**	69.11	96.07	**69.70**
Average	–	52.20	61.83	**69.16**	**264.94**	62.11	214.47

D.3 Data Efficiency

To examine whether CopyGen remains effective under limited training data, we vary the amount of plugin training data on Qwen3-8B in the frozen-model setting, using 50% and 10% of the full training set. Table 7 shows that CopyGen remains clearly better than the base model even under substantially reduced supervision. Using only 10% of the training data still yields clear gains on all four domains, while 50% is already sufficient to recover most of the full-data performance and even surpass it in some cases. These results show that CopyGen remains effective with relatively limited training data, while additional supervision is most helpful on the more challenging settings.

Table 7: Effect of training data size on Qwen3-8B with CopyGen. Best results in **bold**.

Method	Contract		Literature		Code		Random Sequence		Average	
	EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)	EM (↑)	Tokens/s (↑)
Base Model	16.31	53.00	77.24	56.30	80.35	56.39	67.20	56.18	60.28	55.47
CopyGen-100%	**34.75**	**219.54**	**86.21**	**592.85**	**96.51**	72.79	**90.40**	2483.23	**76.97**	842.10
CopyGen-50%	**34.75**	**219.54**	85.52	480.51	96.07	**73.36**	**90.40**	2483.23	76.69	814.16
CopyGen-10%	21.99	166.21	84.83	564.28	89.08	55.35	87.60	**2661.27**	70.88	**861.78**

D.4 Learned Calibration versus Fixed Margin Rules

To isolate what the learned bias head contributes beyond a simple rule, we replace it with two prespecified heuristics applied after five grounded tokens. The first scales the raw margin by 2, and the second adds a constant 1, which is not sign-equivalent to the raw margin under $\gamma = 0$. The trigger and length heads are kept unchanged, and all experiments use Qwen3-8B with temperature 0. As shown in Table 8, both rules yield only marginal gains over the base model, whereas the learned bias head improves average EM on the real-world domains by nearly 15 points. This is because mid-span drifts and true copy boundaries overlap in raw-margin range, so a fixed rule shifts both in the same direction, while distinguishing them requires the hidden-state context used by the bias head.

Table 8: Learned calibration versus fixed margin rules on Qwen3-8B. Avg is computed over the three real-world domains. Best results are highlighted in **bold**.

Calibration	Contract	Literature	Code	Avg	Random Sequence
Base Model	16.31	77.24	80.35	57.97	67.20
Raw margin $\times 2$	16.31	77.24	80.79	58.11	67.60
Raw margin $+1$	17.02	77.93	83.41	59.45	71.20
Learned bias head (CopyGen)	**35.46**	**86.21**	**96.94**	**72.87**	**90.40**

D.5 Robustness to Prompts and Decoding

We examine whether the gains of CopyGen depend on the evaluation template or the decoding strategy. All experiments in this subsection use Qwen3-8B and reuse the trained checkpoint of each domain without retraining.

Prompt and output-format robustness. We modify only the test-time prompt, using two paraphrases with different wording and sentence structure (PA, PB) and a tagged-output format (Tag) that requires the answer to be wrapped in `<COPYGEN_OUTPUT>` tags. As shown in Table 9, CopyGen consistently outperforms the base model under all nine unseen variants, with an average gain of $+13.04$ EM. On *Contract* PA, the gain even widens from $+18.44$ under the original template to $+22.70$, which would not be expected if the learned heads were overfitted to a particular template.

Table 9: Robustness to unseen instruction paraphrases (PA, PB) and output format (Tag) on Qwen3-8B. *Orig.* denotes the original evaluation template in Table 1. Avg is computed over the nine unseen variants. Best results are highlighted in **bold**.

Method	Contract				Literature				Code				Avg
	Orig.	PA	PB	Tag	*Orig.*	PA	PB	Tag	*Orig.*	PA	PB	Tag	
Base Model	16.31	14.89	16.31	16.31	77.24	77.93	55.86	73.79	80.35	71.18	89.52	89.08	56.10
CopyGen	**34.75**	**37.59**	**34.04**	**33.33**	**86.21**	**86.21**	**75.17**	**78.62**	**96.51**	**84.72**	**97.38**	**95.20**	**69.14**
ΔEM	+18.44	+22.70	+17.73	+17.02	+8.97	+8.28	+19.31	+4.83	+16.16	+13.54	+7.86	+6.11	+13.04

Non-greedy decoding. We evaluate at temperature 0.7 with top-p=0.9 or min-p=0.05 on the three real-world domains using paired seeds 42/43/44. Verification remains a deterministic margin decision, and sampling applies to ordinary generation and rejected-proposal fallback. As shown in Table 10, CopyGen exceeds the base model under every sampler and seed, and all values stay close to the greedy results in Table 1.

Table 10: Exact match under non-greedy decoding on Qwen3-8B (temperature 0.7). SD denotes standard deviation over seeds.

Domain	Sampler	Base (s42/s43/s44)	CopyGen (s42/s43/s44)	Gain ± SD
Contract	top-p	16.31/16.31/17.02	35.46/36.17/34.04	+18.68±1.48
	min-p	16.31/16.31/17.02	35.46/36.17/34.75	+18.91±1.08
Literature	top-p	76.55/76.55/75.86	86.90/86.21/87.59	+10.57±1.05
	min-p	76.55/76.55/76.55	86.90/86.21/87.59	+10.34±0.69
Code	top-p	80.79/80.35/81.22	96.94/96.07/96.07	+15.57±0.67
	min-p	80.79/81.22/80.79	96.07/96.51/96.07	+15.28±0.00

Distribution distortion at and after copy exit. We further check whether the calibration head distorts the sampling distribution around the copy boundary. We teacher-force each example to the end of its gold answer so that the base model and CopyGen reach an identical state, and compare their next-token distributions under both samplers. As shown in Table 11, the distributions are nearly identical, with a mean total variation below 0.002 and no top-1 flips. To test whether distortion persists after exit, we append an instruction requesting a short, varied free continuation, teacher-force the correct closing bridge, and sample three seeds per task. Table 12 shows that all diversity statistics are identical between the two models, since the plugin heads are never invoked after a correct exit.

Table 11: Next-token distribution distortion at the copy exit boundary. TV denotes total variation distance between base and CopyGen distributions (0 means identical).

Probe	Comparable states	Top-1 flips	Mean TV	p95 TV
Gold-answer boundary (3 real domains)	98.4%	0 (3 after masking EOS)	0.000285	0
First actual exit (Code)	95.6%	0	0.001904	0

Table 12: Post-exit sampling diversity. All statistics are identical between the base model and CopyGen; higher diversity values indicate more varied continuations across seeds.

Sampler	Token-identical pairs	Heads invoked	Unique rate	Pairwise diff.	Token-Jaccard dist.
top-p	100%	0	0.8750	0.8542	0.3177
min-p	100%	0	0.9583	0.9479	0.3845

D.6 Integration with Inference Engines

We integrate CopyGen with Qwen3-8B into vLLM v0.25.1 [12] and SGLang v0.5.15.post1 [30]. In vLLM, the engine executes the target model while the control logic of CopyGen runs outside it. In SGLang, CopyGen is registered as a custom `SpeculativeAlgorithm` inside the native speculative scheduler, so that no client-side re-prefill is needed. All measurements use batch size 8 on a single RTX PRO 6000 GPU, and Table 13 reports per-domain results. To bound the cost of the plugin heads, we run each example twice on an identical token trajectory, once normally and once with all three heads forced on every decoding step. The difference accounts for at most 1.746% of wall-clock time in vLLM and 2.696% in SGLang, since the speedup of CopyGen comes from reducing the number of decoding steps rather than making each step faster.

E Benchmark Details

We construct four benchmark domains to evaluate exact span reproduction under realistic conditions. Together, they cover settings where even small deviations in copied text can make the output unusable: legal clauses whose wording and numbering must remain intact, scientific evidence that must be quoted verbatim, code spans whose formatting and symbols are execution-critical, and low-redundancy strings that cannot be recovered from semantics alone.

Table 13: End-to-end evaluation of CopyGen in vLLM and SGLang on Qwen3-8B. Average speedup is the mean of per-domain speedups. Better results are highlighted in **bold**.

Domain	EM (↑)			Tokens/s (↑)		
	Base	CopyGen	Δ	Base	CopyGen	Speedup
vLLM						
Contract	16.31	**34.75**	+18.44	381.94	**1294.97**	3.39×
Literature	77.24	**85.52**	+8.28	172.24	**838.85**	4.87×
Code	80.79	**96.94**	+16.15	448.13	**518.45**	1.16×
Random	68.40	**90.80**	+22.40	495.95	**6718.40**	13.55×
Avg (3 real domains)	58.11	**72.40**	+14.29	334.10	**884.09**	3.14×
Avg (all 4 domains)	60.69	**77.00**	+16.31	374.57	**2342.67**	5.74×
SGLang						
Contract	16.31	**35.46**	+19.15	388.36	**672.91**	1.73×
Literature	77.24	**85.52**	+8.28	317.03	**1667.93**	5.26×
Code	79.91	**96.07**	+16.16	392.61	**476.75**	1.21×
Random	69.20	**91.20**	+22.00	367.86	**5480.38**	14.90×
Avg (3 real domains)	57.82	**72.35**	+14.53	366.00	**939.20**	2.74×
Avg (all 4 domains)	60.67	**77.06**	+16.39	366.47	**2074.49**	5.78×

Contracts. In the contract domain, we focus on structured *legal clause extraction*. We collected 2,662 exhibit files from SEC EDGAR filings[2] and extracted text from 2,468 of them. Given a local contract window, the model extracts requested clauses and returns them in a JSON-style object. The output must preserve the original legal text exactly, including clause headings, numbering, and enumerated subclauses. This reflects legal drafting and review workflows where downstream systems need exact clause text rather than paraphrased summaries.

Literature. In the literature domain, we focus on *scientific evidence extraction*. We curated 868 full-text articles from PubMed Central (PMC)[3], covering 13,206 article sections. Given a local article context, the model returns the exact evidence span for a requested scientific detail, such as an experimental protocol, sample description, assay procedure, or statistical analysis statement. This reflects scientific search and review workflows where users need faithful quotations from the original paper, since small wording changes can distort methodological or empirical claims.

Code. In the code domain, we focus on *code explanation*, using several well-known open-source Python web and API repositories, including `FastAPI`, `SQLModel`, and `Starlette`, whose APIs and test code is realistic and standard, allowing us to focus on faithful reproduction rather than deep program reasoning. Together, these repositories contain over 1,500 Python files, more than 130K non-blank lines of Python code, and 766 test files. Given a local code context, the model briefly explains a target snippet and then reproduces it exactly. The task emphasizes faithful reproduction rather than deep program reasoning, since many target snippets contain fragile structures such as nested dictionaries, string literals, and brackets.

Random Sequence. We additionally include an *exact long-string reproduction* task to isolate faithful copy when the target has little semantic redundancy. Specifically, we generate realistic OpenSSH-style private-key files and ask the model to reproduce the key sequence exactly. All values are generated for this study and do not correspond to real credentials. The target strings are long, low-redundancy sequences containing line breaks, mixed-case letters, digits, and symbols such as / and +, reflecting software and infrastructure workflows where encoded configuration values, serialized artifacts, or credential-like identifiers must be copied exactly.

Table 14 summarizes the domains, tasks, and corresponding train/test sizes.

[2] `https://www.sec.gov/edgar/search/`
[3] `https://pmc.ncbi.nlm.nih.gov/`

Table 14: Dataset statistics for each domain and task.

Domain	Task	# Train	# Test	# Total
Contract	Legal Clause Extraction	658	141	799
Literature	Scientific Evidence Extraction	712	145	857
Code	Code Explanation	821	229	1050
Random Sequence	Exact Long-String Reproduction	2250	250	2500

F Proposal Verification

For a candidate source start $s \in \mathcal{C}_t$, let $\pi_t^{(s)} = x_{s:s+\hat{\ell}_t-1}$ denote the proposed source span. We replay this proposal token by token with the frozen base model. For the k-th proposal token, let $o_{t,k}^{(s)} \in \mathbb{R}^{|\mathcal{V}|}$ be the replay logit vector, and let $\pi_{t,k}^{(s)}$ denote the proposed source token at that position. We define the raw source margin as

$$\Delta_{t,k}^{(s)} = o_{t,k}^{(s)}[\pi_{t,k}^{(s)}] - \max_{v \in \mathcal{V},\, v \neq \pi_{t,k}^{(s)}} o_{t,k}^{(s)}[v]. \tag{8}$$

This margin measures how strongly the frozen model prefers the proposed source token over the strongest alternative at the same decoding step.

We then apply a lightweight bias head to calibrate this raw margin. Specifically, a bias head $f_{\text{bias}}(\cdot)$ computes

$$b_{t,k}^{(s)} = \alpha \tanh\Big(f_{\text{bias}}\big(\text{LN}([\bar{h}_{t,k}^{(s)}; \Delta_{t,k}^{(s)}])\big)\Big),$$

where $\bar{h}_{t,k}^{(s)}$ is the replay hidden state at the k-th proposal position and α bounds the correction magnitude. The adjusted verifier margin is then

$$\widetilde{\Delta}_{t,k}^{(s)} = \Delta_{t,k}^{(s)} + b_{t,k}^{(s)}.$$

Intuitively, the bias head learns when the source token should be reinforced and when it should be suppressed. For internal mismatch cases, it can push an under-ranked but correct source token above its competitor. For overrun cases, it can push a marginally favored source continuation token back below the alternative that should be generated instead.

A proposal is accepted only up to its longest verified prefix. We first define the accepted length as

$$a_t^{(s)} = \max\Big\{L \in \{0, \ldots, |\pi_t^{(s)}|\} : \widetilde{\Delta}_{t,k}^{(s)} > \gamma,\ \forall k = 1, \ldots, L\Big\},$$

where γ is the verifier threshold, and the accepted prefix is $A_t^{(s)} = \pi_t^{(s)}[1 : a_t^{(s)}]$. Thus, the decoder emits only the longest safe prefix and truncates the proposal at the first unsafe token. Among all candidates, we rank proposals primarily by accepted length, and use cumulative adjusted margin and cumulative log-probability as tie-breakers.

If the best candidate $s^\star$ satisfies $a_t^{(s^\star)} > 0$, the accepted prefix $A_t^{(s^\star)}$ is emitted in one step. Otherwise, the decoder performs a one-step fallback generation. This fallback is also calibrated by the same bias mechanism rather than reverting to plain greedy decoding. Let

$$\mathcal{V}_t^{\text{src}} = \{x_j : j \in \mathcal{C}_t\}$$

denote the unique next-source tokens supported by the currently aligned continuation set. For each $v \in \mathcal{V}_t^{\text{src}}$, we define a fallback raw margin

$$\Delta_t^{\text{fb}}(v) = o_t[v] - \max_{u \in \mathcal{V},\, u \neq v} o_t[u],$$

where o_t is the frozen model's next-token logit vector at the current decoding step. We reuse the same bias head to obtain

$$b_t^{\text{fb}}(v) = \alpha \tanh\Big(f_{\text{bias}}\big(\text{LN}([h_t; \Delta_t^{\text{fb}}(v)])\big)\Big),$$

and form calibrated fallback logits

$$\bar{o}_t[v] = \begin{cases} o_t[v] + b_t^{\mathrm{fb}}(v), & v \in \mathcal{V}_t^{\mathrm{src}}, \\ o_t[v], & \text{otherwise.} \end{cases}$$

This makes fallback generation more source-aware: when the correct next token should still follow the source, it can be strengthened; when source-following should end, candidate source-continuation tokens can be suppressed.

We train the bias head using token-level verifier supervision. For each replayed verifier position i, let $y_i \in \{+1, -1\}$ indicate whether the proposal token is a positive in-span copy token or a negative boundary-crossing token, and let Δ_i^{raw} and Δ_i^{adj} denote its raw and adjusted margins. Since most verifier decisions are easy and far from the decision boundary, we focus learning on the ambiguous cases by defining a hardness weight

$$w_i^{\mathrm{hard}} = \sigma\left(-\frac{y_i \Delta_i^{\mathrm{raw}}}{\tau_w}\right),$$

where $\sigma(\cdot)$ denotes the logistic sigmoid and τ_w is a temperature parameter. Hard cases near the raw decision boundary receive larger weight, while easy high-confidence cases receive smaller weight.

The main verifier objective is

$$\mathcal{L}_{\mathrm{margin}} = \frac{\sum_{i:y_i=+1} w_i^{\mathrm{hard}} [m_+ - \Delta_i^{\mathrm{adj}}]_+}{\sum_{i:y_i=+1} w_i^{\mathrm{hard}} + \varepsilon} + \lambda_- \frac{\sum_{i:y_i=-1} w_i^{\mathrm{hard}} [\Delta_i^{\mathrm{adj}} + m_-]_+}{\sum_{i:y_i=-1} w_i^{\mathrm{hard}} + \varepsilon},$$

where $m_+ > 0$ and $m_- > 0$ are safety margins for positive and negative verifier decisions, respectively, $[z]_+ = \max(z, 0)$, λ_- weights the negative term, and ε is a small constant for numerical stability. This objective encourages positive copy tokens to remain safely above the decision boundary and negative boundary-crossing tokens to remain below it.

The final objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{trig}} + \lambda_{\mathrm{len}} \mathcal{L}_{\mathrm{len}} + \lambda_{\mathrm{bias}} \mathcal{L}_{\mathrm{margin}},$$

where $\mathcal{L}_{\mathrm{trig}}$ and $\mathcal{L}_{\mathrm{len}}$ denote the trigger and span-length objectives, and λ_{len} and λ_{bias} weight the length and margin terms, respectively. Overall, the verifier turns copy acceptance into a calibrated decision process: high-confidence source tokens pass through unchanged, difficult in-span tokens can be rescued by a positive correction, and marginal boundary-crossing tokens can be suppressed before they trigger over-copy.

G Implementation Details

For the LoRA baseline, we insert LoRA adapters into the attention projection modules of the base model, namely `q_proj`, `k_proj`, `v_proj`, and `o_proj`. We use LoRA with rank 16, scaling factor 32, and dropout 0.05. Training runs for one epoch with AdamW, a learning rate of 2×10^{-4}, gradient accumulation of 16, and a warmup ratio of 0.1. Following standard instruction-tuning practice, the loss is computed only on the target output, with prompt tokens masked out. At inference time, all baselines use greedy decoding.

For CopyGen, we freeze the backbone LM and train only the lightweight trigger, length, and verifier-calibration heads. Each head is implemented as a small multilayer perceptron over decoder hidden states, with LayerNorm, GELU activations, and dropout. Training runs for one epoch with AdamW, a learning rate of 10^{-4}, a cosine learning-rate schedule, and gradient accumulation of 16. The training objective combines trigger prediction, span-length prediction, and verifier calibration, with $\lambda_{\mathrm{len}} = 1$ and $\lambda_{\mathrm{bias}} = 1$. Copy supervision is activated only when the current prefix has sufficient source grounding, using $\tau = 5$, and the length objective is applied to the tentative span prediction. For verifier calibration, we use $m_+ = m_- = 1$, $\lambda_- = 1$, hardness temperature $\tau_w = 1$, numerical constant $\varepsilon = 10^{-8}$, and maximum bias magnitude $\alpha = 8$. At inference time, CopyGen uses deterministic decoding with temperature set to 0. Proposal verification uses a verifier threshold of $\gamma = 0$, and when no proposal is accepted, decoding falls back to a source-aware one-step generation rule.

Throughput is measured with a single inference process on an otherwise idle GPU, so that the reported decoding speed is not affected by concurrent workloads.

Before computing exact match, we apply the same preprocessing procedure to both model outputs and reference answers to remove harmless formatting differences rather than relax the correctness criterion. For all tasks, this preprocessing strips leading and trailing whitespace and removes empty lines. For the code task, we additionally ignore whitespace-only formatting differences, such as uniform global indentation shifts and extra or missing blank lines. All remaining token, boundary, and content mismatches are counted as errors.

H Full Experimental Results

H.1 Full Baseline Comparisons

Table 15 reports per-backbone results for all baselines in Table 2. CopyGen achieves the best average EM and throughput on every backbone, and consistently obtains the highest EM on *Contract*, *Literature*, and *Random Sequence*.

Table 15: Full comparison with source-reuse and constrained-decoding baselines. Baselines are grouped into lossless source-reuse acceleration (*Lossless*), rule-based source constraints (*Rule-based*), and learned copy mechanisms (*Learned*); CopyGen is listed separately. Each cell reports EM and tokens/s. Throughput of Longest-Copy is omitted (–) as it is not meaningful without model verification. Best results within each backbone are highlighted in **bold**.

Type	Method	Contract		Literature		Code		Random Sequence		Average	
		EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s
						Qwen3-4B					
Base Model		5.67	66.5	68.28	60.1	54.15	54.8	53.60	62.3	45.43	60.9
Lossless	LLMA	5.67	**260.9**	68.28	517.1	54.15	139.6	53.60	223.4	45.43	285.3
	CopySpec	5.67	187.2	68.28	313.8	54.15	136.7	53.60	116.3	45.43	188.5
Rule-based	PICARD-style	0.00	52.0	67.59	52.5	84.28	50.2	53.60	231.0	51.37	96.4
	Longest-Copy	0.00	–	12.41	–	**86.46**	–	0.00	–	24.72	–
Learned	CopySpan	0.71	54.5	10.34	**741.8**	82.97	**440.7**	1.60	2522.2	23.91	939.8
CopyGen (Ours)		**13.48**	238.0	**84.83**	729.4	55.02	132.3	**74.80**	**2939.2**	**57.03**	**1009.7**
						Qwen3-8B					
Base Model		16.31	53.0	77.24	56.3	80.35	56.4	67.20	56.2	60.28	55.5
Lossless	LLMA	16.31	**234.6**	77.24	493.7	80.35	94.8	67.20	588.0	60.28	352.8
	CopySpec	16.31	172.8	77.24	300.4	80.35	**96.3**	67.20	334.2	60.28	225.9
Rule-based	PICARD-style	0.00	50.8	77.93	51.3	96.07	48.8	69.20	227.4	60.80	94.6
	Longest-Copy	0.00	–	12.41	–	95.63	–	0.00	–	27.01	–
Learned	CopySpan	7.09	27.0	28.97	102.2	79.04	54.4	5.60	1335.3	30.18	379.7
CopyGen (Ours)		**34.75**	219.5	**86.21**	**592.9**	**96.51**	72.8	**90.40**	**2483.2**	**76.97**	**842.1**
						Ministral-8B					
Base Model		19.15	60.0	41.38	62.1	96.07	63.4	42.80	60.9	49.85	61.6
Lossless	LLMA	19.15	**220.3**	41.38	469.4	96.07	94.7	42.80	53.4	49.85	209.5
	CopySpec	19.15	191.4	41.38	289.5	96.07	102.3	42.80	64.8	49.85	162.0
Rule-based	PICARD-style	0.00	60.0	42.07	60.4	87.77	59.2	40.80	82.1	42.66	65.4
	Longest-Copy	0.00	–	12.41	–	**96.51**	–	0.00	–	27.23	–
Learned	CopySpan	2.13	58.9	19.31	415.7	93.01	**514.3**	54.00	410.9	42.11	349.9
CopyGen (Ours)		**22.70**	204.2	**88.28**	**521.5**	**96.51**	69.1	**100.00**	**2609.6**	**76.87**	**851.1**

H.2 Evaluation of Frontier LLMs

Table 16 compares three frontier LLMs with the open backbones under the same evaluation protocol, where the frontier LLMs are accessed through their official APIs. Results show that even the strongest frontier LLM falls well short of perfect exact copying.

H.3 Ablation Study

Table 17 reports the full ablation results across all models and domains. The results show that all three components contribute to the final performance: adaptive length proposal improves the speed–accuracy tradeoff, adaptive weighting improves verifier learning, and margin calibration is critical for reliable copy control.

Table 16: Exact match of frontier LLMs and open backbones with and without CopyGen under the same evaluation protocol. Best results in each row are highlighted in **bold**.

Domain	Frontier LLMs			Qwen3-4B			Qwen3-8B			Ministral-8B		
	GPT-4.1 -nano	Gemini-2.5 -Flash-Lite	DeepSeek -V4-Pro	Base	CopyGen	$\text{CopyGen}_{\text{SFT}}$	Base	CopyGen	$\text{CopyGen}_{\text{SFT}}$	Base	CopyGen	$\text{CopyGen}_{\text{SFT}}$
Contract	33.33	41.84	56.03	5.67	13.48	90.78	16.31	34.75	96.45	19.15	22.70	**98.58**
Literature	80.00	80.00	77.24	68.28	84.83	85.52	77.24	86.21	**94.48**	41.38	88.28	92.41
Code	65.50	90.83	97.38	54.15	55.02	98.69	80.35	96.51	98.25	96.07	96.51	**100.00**
Random	81.20	78.40	99.20	53.60	74.80	90.40	67.20	90.40	96.80	42.80	**100.00**	**100.00**
Avg	65.01	72.77	82.46	45.43	57.03	91.35	60.28	76.97	96.50	49.85	76.87	**97.75**

Table 17: Full ablation results of CopyGen. For *w/o Length Proposal*, we replace the adaptive span proposal with fixed copy lengths $k \in \{8, 16, 32, 64, 128, 256\}$ and report the average over the sweep. Average is computed over all four domains. Best results are highlighted in **bold**.

Model	Variant	Contract		Literature		Code		Random Sequence		Average	
		EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s	EM	Tokens/s
Qwen3-4B	**Full**	**13.48**	**238.02**	**84.83**	**729.42**	**55.02**	**132.28**	74.80	**2939.15**	**57.03**	**1009.72**
	w/o Length Proposal	13.01	206.44	84.02	474.31	54.88	118.72	**74.87**	1010.51	56.69	452.49
	w/o Adaptive Weighting	12.77	223.38	68.28	606.32	53.28	123.41	52.80	1809.08	46.78	690.55
	w/o Margin Calibration	5.67	235.28	67.59	636.08	53.28	123.12	53.20	2198.09	44.94	798.14
Qwen3-8B	**Full**	**34.75**	**219.54**	**86.21**	**592.85**	**96.51**	**72.79**	**90.40**	**2483.23**	**76.97**	**842.10**
	w/o Length Proposal	29.79	173.06	**86.21**	379.71	94.47	69.78	86.80	797.19	74.32	354.94
	w/o Adaptive Weighting	29.79	194.64	76.55	530.84	80.35	71.24	68.00	1828.15	63.67	656.22
	w/o Margin Calibration	16.31	205.20	77.24	509.79	79.91	69.73	68.40	1895.75	60.47	670.12
Ministral-8B	**Full**	**22.70**	**204.22**	**88.28**	**521.48**	**96.51**	**69.11**	**100.00**	**2609.59**	**76.87**	**851.10**
	w/o Length Proposal	21.99	184.11	88.17	407.61	**96.51**	65.85	93.13	538.21	74.95	298.94
	w/o Adaptive Weighting	19.86	211.49	64.83	463.85	96.07	67.76	62.80	426.87	60.89	292.49
	w/o Margin Calibration	19.86	213.13	41.38	396.20	96.07	67.48	42.80	296.72	50.03	243.38

H.4 Copy Dynamics Analysis

We provide the full results of the copy dynamics analysis, as shown in Figures 7, 8, and 9. Across models and domains, the proposed lengths are often longer than the accepted lengths, showing that the length predictor provides tentative spans while the verifier determines the final copy boundary. The bucket-level acceptance plots further show that longer proposed spans are not always fully accepted, highlighting the role of verification in preventing over-copy.

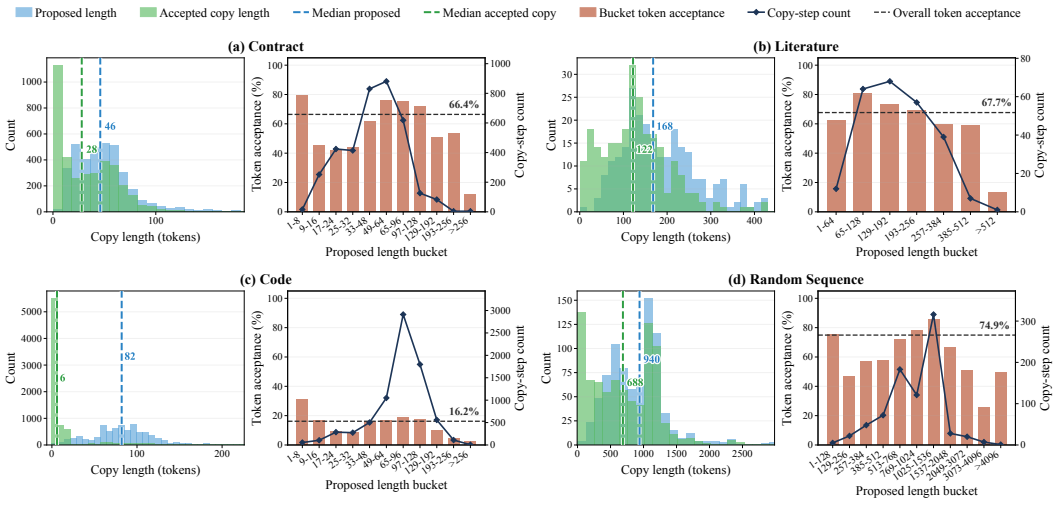

Figure 7: Copy dynamics for Qwen3-4B across four benchmark domains. For each domain, we show the distribution of proposed and accepted copy lengths (left), together with token acceptance by proposed-length bucket (right).

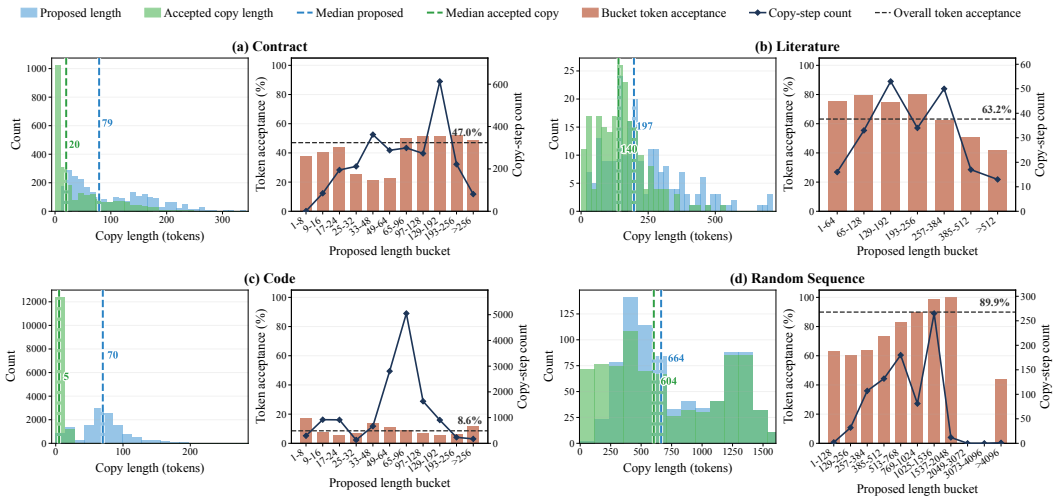

Figure 8: Copy dynamics for Qwen3-8B across four benchmark domains. For each domain, we show the distribution of proposed and accepted copy lengths (left), together with token acceptance by proposed-length bucket (right).

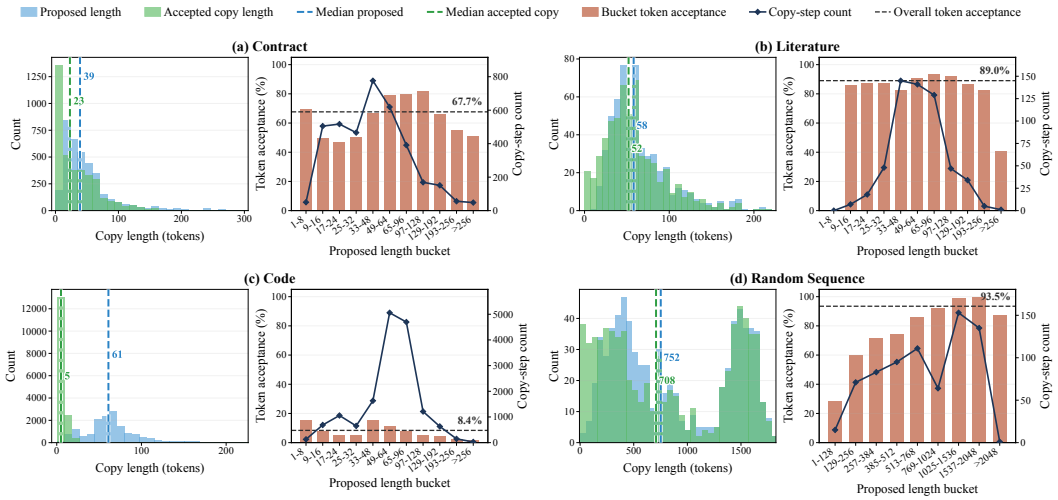

Figure 9: Copy dynamics for Ministral-8B across four benchmark domains. For each domain, we show the distribution of proposed and accepted copy lengths (left), together with token acceptance by proposed-length bucket (right).

H.5 Training Efficiency Analysis

Table 18 reports the training resource usage of CopyGen compared with SFT. Across models and domains, CopyGen substantially reduces peak GPU memory while maintaining comparable average training time per example.

I Licenses and Asset Usage

We document the external assets used in this work, including model backbones, data sources, and code repositories, together with their licenses or terms of use.

Model Backbones. We evaluate `Qwen3-4B` and `Qwen3-8B`, released by Qwen under the Apache 2.0 License and available at `https://huggingface.co/Qwen/Qwen3-4B` and `https://huggingface.co/Qwen/Qwen3-8B`. We also evaluate `Ministral-8B-Instruct`, released by

Table 18: Training resource comparison. We report peak GPU memory during training (GiB) and average training time per example (s/item). Relative changes in parentheses are computed against SFT within each model and task. Lower is better for both metrics.

Method	Contract		Literature		Code		Random Sequence		Average	
	Peak GiB	Train s/item	Peak GiB	Train s/item	Peak GiB	Train s/item	Peak GiB	Train s/item	Peak GiB	Train s/item
Qwen3-4B										
SFT	27.62	**1.316**	21.28	2.610	18.43	2.749	41.67	**1.064**	27.25	1.935
CopyGen	**11.04** (-60.0%)	1.640 (+24.6%)	**10.53** (-50.5%)	**0.625** (-76.1%)	**9.67** (-47.5%)	2.500 (-9.1%)	**12.57** (-69.8%)	1.667 (+56.6%)	**10.95** (-59.8%)	**1.608** (-16.9%)
CopyGen$_{\text{SFT}}$	11.08 (-59.9%)	1.800 (+36.8%)	10.56 (-50.4%)	1.667 (-36.1%)	9.72 (-47.3%)	**2.000** (-27.2%)	12.63 (-69.7%)	1.429 (+34.3%)	11.00 (-59.6%)	1.724 (-10.9%)
Qwen3-8B										
SFT	36.31	**1.897**	29.40	1.795	26.45	1.967	50.05	**0.729**	35.55	1.597
CopyGen	**19.19** (-47.1%)	2.370 (+24.9%)	**18.47** (-37.2%)	**1.250** (-30.4%)	**17.67** (-33.2%)	1.667 (-15.3%)	**20.70** (-58.6%)	1.111 (+52.4%)	**19.01** (-46.5%)	1.600 (+0.2%)
CopyGen$_{\text{SFT}}$	19.24 (-47.0%)	2.680 (+41.2%)	18.49 (-37.1%)	**1.250** (-30.4%)	17.74 (-32.9%)	**1.429** (-27.4%)	20.76 (-58.5%)	1.000 (+37.2%)	19.06 (-46.4%)	**1.590** (-0.5%)
Ministral-8B										
SFT	32.02	**1.748**	26.74	2.060	24.92	2.029	46.34	**0.718**	32.51	1.639
CopyGen	**19.01** (-40.6%)	2.390 (+36.7%)	**17.87** (-33.2%)	1.429 (-30.7%)	**17.44** (-30.0%)	1.667 (-17.9%)	20.27 (-56.3%)	1.111 (+54.7%)	**18.65** (-42.6%)	1.649 (+0.6%)
CopyGen$_{\text{SFT}}$	19.07 (-40.4%)	2.720 (+55.6%)	17.91 (-33.0%)	**1.250** (-39.3%)	17.52 (-29.7%)	**1.429** (-29.6%)	**20.22** (-56.4%)	1.000 (+39.3%)	18.68 (-42.5%)	**1.600** (-2.4%)

Mistral AI under the Mistral Research License and available at `https://huggingface.co/mistralai/Ministral-8B-Instruct-2410`. All model backbones are used for research evaluation in accordance with their corresponding licenses and model cards.

Contract Data. The contract task is constructed from publicly accessible SEC EDGAR filings, available through the SEC EDGAR system at `https://www.sec.gov/edgar/search/`. We use these filings for research purposes and follow the SEC access policies, including fair-access requirements for automated retrieval.

Scientific Literature Data. The scientific-literature task is constructed from full-text articles in the PubMed Central Open Access subset, available at `https://pmc.ncbi.nlm.nih.gov/`. Since PMC Open Access articles are distributed under article-level licenses, we follow the license terms associated with each article.

Code Data. The code task is constructed from publicly available open-source Python repositories, including `FastAPI`, `SQLModel`, and `Starlette`. `FastAPI` and `SQLModel` are released under the MIT License, and `Starlette` is released under the BSD 3-Clause License. These repositories are available at `https://github.com/fastapi/fastapi`, `https://github.com/fastapi/sqlmodel`, and `https://github.com/Kludex/starlette`.

Random Sequences. The random-sequence task is generated by us. Although the strings are formatted to resemble credential-like or encoded configuration values, all such strings are synthetic and do not correspond to real credentials, secrets, or private keys.

All external assets are credited to their original creators or maintainers, with source URLs and license information provided above where applicable. We use these assets only for research and evaluation purposes and in compliance with their corresponding licenses, model cards, access policies, or terms of use. We do not use proprietary, private, or restricted-access data in this study.

J Prompt Templates

Code

````
System:
You are a precise long-context text assistant. Follow the user's instruction over the
     provided source text. When the instruction asks for exact copying or quoting,
     preserve the source wording exactly. Output only the requested final answer. Do not
     add explanations, notes, markdown fences, or any extra content.

User:
Instruction:

In the Python function `<function_name>`, briefly explain what the anchored block is used
      for, then quote exactly the block beginning with `<anchor_line>`, ending at the
     matching closing line for that block. Preserve every line, indentation level, quote,
      comma, bracket, and any trailing comma on the closing line.

Source text:

[FILE]
<file_name>

[LINES]
<start_line>-<end_line>

````python
<local_python_context>
Constraint:
Return only the requested answer. Nothing else.
````
````

Literature

```
System:
You are a precise long-context text assistant. Follow the user's instruction over the
     provided source text. When the instruction asks for exact copying or quoting,
     preserve the source wording exactly. Output only the requested final answer. Do not
     add explanations, notes, markdown fences, or any extra content.

User:
Instruction:

In the paper "<paper_title>", return the exact content from "<section_title>" that begins
      with "<anchor_text>" and provides the requested scientific detail. Output only the
     copied text.

Source text:

<paper_title>

<section_context>

Constraint:
Return only the requested answer. Nothing else.
```

Contract

```
System:
You are a precise long-context text assistant. Follow the user's instruction over the
     provided source text. When the instruction asks for exact copying or quoting,
     preserve the source wording exactly. Output only the requested final answer. Do not
     add explanations, notes, markdown fences, or any extra content.

User:
Instruction:

Extract the full text of the following clauses exactly as written in the article below:
1. the clause headed "<clause_heading_1>".
2. the clause headed "<clause_heading_2>".
...

Return a JSON object with keys "<clause_key_1>", "<clause_key_2>", ... . Include each
     section heading and preserve the original wording, line breaks, and indentation.
     Output only the JSON object.

Source text:

<contract_article_context>

Constraint:
Return only the requested answer. Nothing else.
```

Random Sequence

```
System:
You are a precise long-context text assistant. Follow the user's instruction over the
     provided source text. When the instruction asks for exact copying or quoting,
     preserve the source wording exactly. Output only the requested final answer. Do not
     add explanations, notes, markdown fences, or any extra content.

User:
Instruction:

Return only the exact string content for the given text. Preserve line breaks exactly as
     written. Output only the copied content.

Source text:

<SOURCE TEXT>

Constraint:
Return only the requested answer. Nothing else.
```

K Task Examples

Here, we show one example for each task type to make the task format concrete.

Contract

Input. Extract the full text of the following clauses exactly as written in the article below: 1. the clause headed “CHOICE OF LAW.” 2. the clause headed “CONSENT TO JURISDICTION.” 3. the clause headed “Entire Agreement.” 4. the clause headed “Survival of Representations.”
Return a JSON object with keys `governing_law_clause`, `consent_to_jurisdiction_clause`, `entire_agreement_clause`, `survival_clause`. Include each section heading and preserve the original wording, line breaks, and indentation. Output only the JSON object.

Source excerpt.

```
ARTICLE X
GENERAL PROVISIONS

10.1. Survival of Representations.
All representations and warranties made hereunder and in any other Loan Document or other document delivered pursuant
      hereto or thereto or in 201380721_5

connection herewith or therewith shall survive the execution and delivery hereof and thereof. Such representations and
       warranties have been or will be relied upon by the Administrative Agent and each Bank ...

...

10.4. Entire Agreement.
This Agreement, together with the other Loan Documents, comprises the complete and integrated agreement of the parties
       on the subject matter hereof and thereof and supersedes all prior agreements, written or oral, on such subject
       matter ...

...

10.12. CHOICE OF LAW.
THE LOAN DOCUMENTS (OTHER THAN THOSE CONTAINING A CONTRARY EXPRESS CHOICE OF LAW PROVISION) SHALL BE CONSTRUED IN
      ACCORDANCE WITH THE INTERNAL LAWS ...

10.13. CONSENT TO JURISDICTION.
ANY LEGAL ACTION, LITIGATION OR PROCEEDING WITH RESPECT TO THIS AGREEMENT OR ANY OTHER LOAN DOCUMENT MAY BE BROUGHT IN
       THE COURTS OF THE STATE OF NEW YORK ...

10.14. WAIVER OF JURY TRIAL.
EACH LOAN PARTY, THE ADMINISTRATIVE AGENT AND EACH BANK HEREBY WAIVE TRIAL BY JURY ...
```

Output.

```
{
  "governing_law_clause": "10.12. CHOICE OF LAW.\nTHE LOAN DOCUMENTS (OTHER THAN THOSE CONTAINING A CONTRARY EXPRESS
       CHOICE OF LAW PROVISION) SHALL BE CONSTRUED IN ACCORDANCE WITH THE INTERNAL LAWS (AND NOT THE LAW OF
       CONFLICTS) OF THE STATE OF NEW YORK BUT GIVING EFFECT TO FEDERAL LAWS APPLICABLE TO NATIONAL BANKS.",
  "consent_to_jurisdiction_clause": "10.13. CONSENT TO JURISDICTION.\nANY LEGAL ACTION, LITIGATION OR PROCEEDING WITH
       RESPECT TO THIS AGREEMENT OR ANY OTHER LOAN DOCUMENT MAY BE BROUGHT IN THE COURTS OF THE STATE OF NEW YORK
       SITTING IN NEW YORK COUNTY OR OF THE UNITED STATES DISTRICT COURT OF THE SOUTHERN DISTRICT OF NEW YORK
       SITTING IN NEW YORK COUNTY, AND ANY APPELLATE COURT FROM ANY THEREOF, AND EACH OF THE PARTIES HERETO
       IRREVOCABLY AND UNCONDITIONALLY SUBMITS TO THE JURISDICTION OF SUCH COURTS AND AGREES THAT ALL CLAIMS IN
       RESPECT OF ANY SUCH ACTION, LITIGATION OR PROCEEDING MAY BE HEARD AND DETERMINED IN SUCH NEW YORK STATE COURT
        OR, TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, IN SUCH FEDERAL COURT. EACH OF THE PARTIES HERETO
       HEREBY IRREVOCABLY WAIVES ANY OBJECTION IT MAY NOW OR HEREAFTER HAVE AS TO THE VENUE OF ANY SUCH SUIT, ACTION
        OR PROCEEDING BROUGHT IN SUCH A COURT OR THAT SUCH COURT IS AN INCONVENIENT FORUM.",
  "entire_agreement_clause": "10.4. Entire Agreement.\nThis Agreement, together with the other Loan Documents,
       comprises the complete and integrated agreement of the parties on the subject matter hereof and thereof and
       supersedes all prior agreements, written or oral, on such subject matter. In the event of any conflict
       between the provisions of this Agreement and those of any other Loan Document, the provisions of this
       Agreement shall control; provided, that the inclusion of supplemental rights or remedies in favor of the
       Administrative Agent or the Banks in any other Loan Document shall not be deemed a conflict with this
       Agreement. Each Loan Document was drafted with the joint participation of the respective parties thereto and
       shall be construed neither against nor in favor of any party, but rather in accordance with the fair meaning
       thereof.",
  "survival_clause": "10.1. Survival of Representations.\nAll representations and warranties made hereunder and in any
        other Loan Document or other document delivered pursuant hereto or thereto or in 201380721_5\n\nconnection
       herewith or therewith shall survive the execution and delivery hereof and thereof. Such representations and
       warranties have been or will be relied upon by the Administrative Agent and each Bank, regardless of any
       investigation made by the Administrative Agent or any Bank or on their behalf and notwithstanding that the
       Administrative Agent or any Bank may have had notice or knowledge of any Default or Unmatured Default at the
       time of any Loan or Commitment, and shall continue in full force and effect as long as any Loan or any other
       Obligation shall remain unpaid or unsatisfied."
}
```

Literature

Input. From the paper *Inhibition of cell growth by EGR-1 in human primary cultures from malignant glioma*, extract the exact content from the subsection headed *DNA Sequencing of p53 Exons 5–9* that contains the sequencing or genotyping procedure and begins with "For analysis of p53 mutations, previously described primer sets were used for PCR amplification from high molecular weight DNA of three DNA fragments corresponding to p53 exons 5 and 6, exon 7, and exons 8 and 9, respectively." Output only the copied text.

Source excerpt. Densitometric analysis of Western blots was performed on a Macintosh G3/233 computer using the public domain NIH Image software. The EGR-1 and fibronectin chemiluminescence signals were normalized with respect to actin content of each sample.

. . .

DNA Sequencing of p53 Exons 5–9. For analysis of p53 mutations, previously described primer sets were used for PCR amplification from high molecular weight DNA of three DNA fragments corresponding to p53 exons 5 and 6, exon 7, and exons 8 and 9, respectively [33]. Direct sequencing of the specific DNA fragments was performed using a Big Dye terminator DNA sequencing kit with the ABI PRISM 377 DNA Sequencer (PE Applied Biosystems Inc., Foster City, CA, USA), according to the manufacturer's instructions. Results were analyzed using the ABI sequencing analysis software.

. . .

Analysis of allele dosage for MDM2 and loss of heterozygosity for p16/ARF. Southern blotting was used for monitoring MDM2 gene amplification and loss of heterozygosity at the p16INK4a/ARF locus.

Output. For analysis of p53 mutations, previously described primer sets were used for PCR amplification from high molecular weight DNA of three DNA fragments corresponding to p53 exons 5 and 6, exon 7, and exons 8 and 9, respectively. Direct sequencing of the specific DNA fragments was performed using a Big Dye terminator DNA sequencing kit with the ABI PRISM 377 DNA Sequencer (PE Applied Biosystems Inc., Foster City, CA, USA), according to the manufacturer's instructions. Results were analyzed using the ABI sequencing analysis software.

Code

Input. In the Python function `test_foo_no_needy`, briefly explain what the anchored block is used for, then quote exactly the block beginning with `"detail":  [` for key `detail`, ending at the matching closing line for that block.

Source excerpt.

```
def test_foo_no_needy(client: TestClient):
    response = client.get("/items/foo?skip=a&limit=b")
    assert response.status_code == 422
    assert response.json() == {
        "detail": [
            {
                "type": "missing",
                "loc": ["query", "needy"],
                "msg": "Field required",
                "input": None,
            },
            {
                "type": "int_parsing",
                "loc": ["query", "skip"],
                "msg": "Input should be a valid integer, unable to parse string as an integer",
                "input": "a",
            },
            {
                "type": "int_parsing",
                "loc": ["query", "limit"],
                "msg": "Input should be a valid integer, unable to parse string as an integer",
                "input": "b",
            },
        ]
    }
```

Output.

````
Explanation:
This block specifies the expected validation errors for a request with a missing required query parameter and two non-
      integer query values.

Quoted block:
````python
       "detail": [
           {
               "type": "missing",
               "loc": ["query", "needy"],
               "msg": "Field required",
               "input": None,
           },
           {
               "type": "int_parsing",
               "loc": ["query", "skip"],
               "msg": "Input should be a valid integer, unable to parse string as an integer",
               "input": "a",
           },
           {
               "type": "int_parsing",
               "loc": ["query", "limit"],
               "msg": "Input should be a valid integer, unable to parse string as an integer",
               "input": "b",
           },
       ]
````

Random Sequence

The random-sequence task uses synthetic key-like strings where every character and line break must be preserved.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAQdqHI9+PedkeGKrzeay8vXCutJJT7hJ9UO0gWwS3adFRpogbQy
+SMlY9zhB9CxS8qRI6QTJMgzmdGplVr9cJRsSLEm3C61btnf2uyaajGLiesOFHRDsRMLP6
dDm4SC1tMWgPHyubKvqWwbf9F5RlwDvBPmPLRkC3jWCcEnVNHwO3AIOUG9NOqPvOntLEGr
GmzYmikeJax2eh72D+Iv8n+iXmDiOnu9EXLjCncRyeF9Jxb14poJ16YIDh9mtS6g94sw5n
YzYCdulqqLKrPKkyHOAsv2quMkHrvARiOuHd9aaeoHipcSHjNJE6GC/cvSJTqZMHDKcPxw
d3RDCK6c+eLDRcyHfGvYYMnOPEYEPfYjGMQZuUTQK8As9scjSnF96EiYcMlUi12S4UULfz
...
VHtWPskfnjud3jKIdCuT6byHo3dSzzHjidub4JNtTrT6NFb21KapuDg2v5DYeV/eTRO1bH
P1N3qpn+yu5RYEtNtWKOTYyY6maGNlBjxeFOzVyDMnRKuO3a93yA23vKhSTSnSvCyghwxo
F4oS9tSsvkg5ItCHlEVi4te8y9jP+2XpMeSvmvXWaRC/tOGzkPs6mSkMm+r14pM5r4vKw0
6bd883
-----END OPENSSH PRIVATE KEY-----
```
```

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: The abstract and Introduction summarize the main claims and scope of our work, and the empirical results in Section 4 support these claims.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Appendix A discusses the main limitations of our work.

   Guidelines:

   - The answer [N/A] 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: [N/A]

Justification: This paper does not include theoretical results or formal proofs.

Guidelines:

- The answer [N/A] 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: Sections 3–4 and appendix G provide the method, benchmark construction, implementation details, and evaluation setup needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] 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 provide a github repository with code, benchmark artifacts, and instructions for reproducing the main experiments, and include it as supplementary material at submission time.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix E provide the data splits, model settings, hyperparameters, optimizer, and decoding details.

   Guidelines:

   - The answer [N/A] 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 significance results are based on a two-tailed paired test with $p < 0.05$.

   Guidelines:

   - The answer [N/A] 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: Section 4 and Appendix H.5 report the GPU type, memory usage, and training-time statistics.

   Guidelines:

   - The answer [N/A] 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: The research uses public or synthetic data and follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: Appendix B discusses both beneficial uses and potential risks of exact copying.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The random-sequence data in our work uses synthetic credential-like strings rather than real secrets.

    Guidelines:

    - The answer [N/A] 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: Appendix I documents the external assets and their licenses or terms of use.

    Guidelines:

    - The answer [N/A] 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: [Yes]

    Justification: Appendix E documents the benchmark construction, data sources, and dataset statistics.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve human-subject research so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is only used only for writing, editing, or formatting purposes.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.