Prompt-Based One-Shot Exact Length-Controlled Generation with LLMs

Anonymous ACL submission

Abstract

Countdown-Aided Prompting for Exact Length (CAPEL) is a prompt-only single-pass method that keeps black-box LLMs within exact word or character budgets by appending visible countdown markers and explicit counting rules. It needs no examples, fine-tuning, or iterative decoding, and we evaluate it on random-text generation, XSUM summarization, MT-Bench-LI instruction following, and the LIFEBench Equal-To track. High-capacity models reach $\geq 95\%$ exact match with negligible deviation on random-text and LIFEBench; XSUM retains strong compliance with modest ROUGE shifts; and on MT-Bench-LI `GPT-4.1` achieves $\sim 75\%$ exact match at roughly a one-point drop in judged quality. We chart these compliance–quality–cost trade-offs and release an evaluator for standardized EM/MAE/MALD reporting, positioning CAPEL as a reproducible one-shot baseline for exact length control.

1 Introduction

Length control is the ability of a text-generation system to terminate at an *exact* number of words, sentences, or characters. It underpins headline writing, one-sentence summaries, constrained classroom exercises, and social-media posts with hard character limits, where any overshoot or undershoot renders the text unusable.

Yet even today's strongest large language models drift under strict limits. Prior work narrows the gap by *modifying the model*: retraining with length-labelled data, adding countdown adapters, or using reinforcement learning penalties to deter over- or under-shooting (Feng et al., 2023). Such edits demand substantial compute and direct weight access, excluding proprietary black boxes, while heuristic decoding (e.g., early stopping plus log-probability rescaling) still fails on long passages.

We take a prompt-side view. Counting is difficult for transformers because self-attention lacks an accumulator, so we *externalize* it with visible countdown markers and deterministic rules. The model then matches a pattern, needing no gradient updates, reward models, or extra decoding passes, and the same suffix transfers to proprietary APIs and leading open-source LLMs.

This paper asks whether such prompt-only control can be a reproducible black-box baseline while keeping the compliance–quality–cost trade-offs explicit. We introduce CAPEL, show where it reaches high-90s exact match (RTG, LIFEBench) versus quality dips (MT-Bench-LI single-answer scores, XSUM ROUGE shifts), and release artefacts for replication. Our contributions are: *Preview of baselines.* On XSUM we additionally evaluate *InstructCMP*, an instruction-only sentence-compression control at the reference length, to situate instruction-priming approaches.

- CAPEL acts as a plug-and-play suffix that keeps high-capacity proprietary and open models within target lengths in a single call (Tables 2, 6).
- We release a shared EM/MAE/MALD evaluator and consistent captions that flag whether MALD is scaled (§4.2).
- We map the compliance–quality–cost trade-offs against draft-then-rewrite, MT-Bench-LI judges, and sparse-marker variants to clarify when single-pass prompting is preferable (Tables 4, 3).

2 Related Work

2.1 Training-Based Approaches for Length Control

Training-based methods adjust model weights to meet length budgets. Hansel (Song et al., 2025) adds countdown tokens so generation stops on cue, whereas L1 (Aggarwal and Welleck, 2025) uses

reinforcement learning to cap chain-of-thought steps. Additional work trains length-aware summarizers (Miculicich et al., 2023; Yu et al., 2021), re-scales EOS loss (Stergiadis et al., 2025), or applies length-conditioned preference optimization (Li et al., 2024a). Constrained RL decoding balances length and quality (Chan et al., 2021). Preference-optimization methods such as DPO can still be verbose, prompting variants that disentangle length from quality, add iterative length regularization (iLR-DPO), desensitize the loss to length (LD-DPO), or condition on response length (Park et al., 2024; Liu et al., 2024a,b; Cai et al., 2025). These weight-level interventions curb verbosity but stop short of exact word budgets and rely on curated preference data or online relabeling.

2.2 Inference-Time Techniques

Length can also be steered without retraining. Instruction-tuned models follow natural-language length cues but remain approximate (Zhou et al., 2023b). BB-MH (Gu et al., 2024) enforces an upper bound by rejecting over-long samples. Prompt engineering attaches length hints for nearly isometric translation (Javorský et al., 2025), draft-then-revise summaries (Jie et al., 2024), or zero-shot templates (Retkowski and Waibel, 2025). These strategies transfer to closed models yet often require multiple iterations to hit strict targets.

Single-pass vs. iterative decoding. InstructCMP (Do et al., 2024) uses length priming (optionally with fine-tuning) for sentence compression but cannot guarantee exact budgets. BB-MH (Gu et al., 2024) performs Metropolis–Hastings resampling to enforce the bound while consuming several proposals; "black-box" refers to API access rather than the algorithm itself. CAPEL instead provides a single-pass prompt scaffold that applies unchanged across summarization, instruction following, and open-ended text, while exposing token and wall-clock costs for reproducibility.

2.3 Evaluation Benchmarks for Controllability

Length-sensitive evaluation is expanding. Length-Controlled AlpacaEval mitigates verbosity bias (Dubois et al., 2024), which also stabilizes Arena-Hard (Li et al., 2024b). IFEval (Zhou et al., 2023a), InFoBench (Qin et al., 2024), CFBench (Zhang et al., 2024a), and LIFEBench (Zhang et al., 2025) probe length or formatting obedience and expose sizeable gaps even in state-of-the-art systems.

2.4 Counting Limitations of Large Language Models

LLMs still struggle to count. GPT-4 falters on letter-level tasks (Shin and Kaneko, 2024; Zhang and He, 2024); the Genius Paradox benchmark quantifies this gap (Xu and Ma, 2025). Theory shows bounded-depth Transformers cannot count unboundedly (Feng et al., 2023), yet chain-of-thought (CoT) prompting lifts them to serial computation, sharply raising accuracy (Wei et al., 2022; Li et al., 2024c; Zhang et al., 2024b). Combining character-level tokenization with CoT pushes letter-counting accuracy from under 10% to 90–98% (Zhang et al., 2024c; Chang and Bisk, 2024). These insights motivate our countdown-marker prompt (§3), which externalizes counting into the visible output.

3 Method: A Plug-and-Play Design

3.1 Countdown-Marker Suffix

Transformer LLMs cannot reliably maintain an internal word counter (Shin and Kaneko, 2024). CAPEL remedies this by appending a short, self-contained **suffix** whose countdown markers transform the output into an external scratch pad, effectively upgrading the computation from constant depth (NC^0) to NC^1 (Feng et al., 2023; Li et al., 2024c) and forcing termination at the requested length. We treat **Countdown-Aided Prompting for Exact Length (CAPEL)** as a *plug-and-play* prompt suffix for exact control, used unchanged across the paper.

3.2 CAPEL for English ($S_N^{\mathbf{en}}$)

CAPEL

Required output format

1. For each integer **k** from **{target_length}** down to **1**:
   - Print the marker <k>.
   - Immediately append **one English token**-letters with optional leading/trailing punctuation (comma, period, apostrophe, hyphen, digits, ...). No spaces inside this token.

2. After the word paired with <1>, write <0> and **stop generating**.
3. **Markers and punctuation are not counted** toward the {target_length}-word total; only the English words themselves count.
4. If you risk running short, do not end with dozens of bare markers. Add fresh, meaningful prose; line breaks or repeated filler do not qualify.

Word-count and ordering rules

- Produce **exactly {target_length} markers** (<{target_length}> ... <1>) and **exactly {target_length} words**.
- The numbers inside successive markers must decrease by **1**—no skips or repeats.
- Two markers may never appear back-to-back without an intervening word.
- No extra text is allowed after <0> (not even a newline).

Correct example (N = 5)

<5>Hello,<4>world!<3>How's <2>everything?<1>Great.<0>

Typical errors to avoid

Early stop ending before <1> or omitting <0>

<5>Quick<4>demo<3>ends<0>

Duplicate markers e.g. two <0>s

<5>Wrong<4>marker<3>again<2>here <1>now<0><0>

Markers-only tail long stretch of bare markers

<7>Starts<6>well<5>then<4>stop. <3><2><1><0>

Counting rules (en/zh/code). After stripping markers (<k>, <0>) we apply deterministic rules: (i) **English**—collapse whitespace to single spaces and count *word-like tokens* via `split(" ")`, so hyphenated spans such as `state-of-the-art` remain one segment. We use this definition for all English budgets and Exact-Match reports, and call out when a benchmark (e.g., LIFEBENCH) uses a different tokenizer for its official scores. (ii) **Chinese**—count Unicode code points after marker removal, treating every visible character equally; (iii) **Code (Python)**—reuse the English rule, so runs such as "\n\n" behave as a single delimiter and A=B counts once.

3.3 Code-Aware Extension for MT-Bench

Certain MT-Bench prompts require literal PYTHON blocks, so we add one extra rule to S_N^{en} for those items:

CAPEL – Code-aware extension

For code sections, reuse the English whitespace rule: collapse spaces/tabs/newlines to a single space and split on that space. Do not apply any line-based counting heuristics.

All other instructions remain unchanged, and Section 5 shows that this one-line tweak preserves exact length while keeping multi-line code faithful.

Relationship to length-aware DPO variants. Length-regularized DPO methods (iLR-DPO, LD-DPO, response-conditioned preference modeling) reduce verbosity by retraining on curated preferences (Liu et al., 2024a,b; Cai et al., 2025; Park et al., 2024) but still stop short of exact word-like-token budgets. CAPEL instead enforces the budget in a single inference call and can serve as a prompt-side companion to models trained with DPO.

4 Experiments

4.1 Experimental Setup

Tasks. We evaluate CAPEL on one in-house dataset plus **three** public benchmarks covering open-ended generation, abstractive summarization, multiturn instruction following, and explicit instruction adherence:

- **Random Text Generation** (*English, Chinese*). An *in-house* corpus for this study; each

prompt requests a coherent passage of *exactly* N *tokens* ($N \in [1, 1000]$ words or characters) with unconstrained topic, isolating length compliance.

- **XSUM** (*English*) (Narayan et al., 2018). Following Hansel (Song et al., 2025), we summarize each article under the reference length plus fixed 5/50/120-word budgets to test content-preserving control.
- **MT-Bench-LI** (*English*). Adopting Yuan et al. (2024), we augment each MT-Bench prompt with three target lengths drawn from first-turn answers by GPT-4 Turbo, Claude 3 Opus, and Mistral Large (OpenAI, 2023; Anthropic, 2024; Mistral AI, 2024). These answers only set the targets (not baselines); each length forms an independent instance, yielding 240 prompts that reduce to 175 unique targets after collapsing duplicates.
- **LIFEBench** (*English, Chinese*) (Zhang et al., 2025). We use the *Equal To* control across ten explicit lengths (16–8192 tokens) and four task families, matching exact-length goals and stress-testing long budgets while avoiding band-style confounds.

Models. We evaluate **11 large language models** spanning three families (Table 1). The three QWEN3 checkpoints are executed locally on a single RTX-A6000, whereas the remaining eight models are invoked through their vendor APIs. Detailed specifications–parameter counts, context windows, maximum completion length, quantisation format, and licence–are provided in Table 8 of Appendix A.

Baselines. We compare CAPEL against (i) **single-pass prompts** and (ii) **iterative resampling**. The global single-pass baseline keeps only the plain instruction ("*Please answer the instruction in exactly ℓ words/characters.*") so we can measure each model's intrinsic length tracking without scaffolds.

For iterative resampling we report BB-MH (Gu et al., 2024) on MT-Bench-LI (§4.5), the setting where it was introduced and validated under a single upper-bound instruction.

Because XSUM's single-sentence summaries closely match sentence compression, we also include **InstructCMP** (Do et al., 2024) as an instruction-only control **restricted to the reference-length setting**. We run it without fine-tuning. Length priming improves adherence but still lacks exact budgets, and extending the method to fixed 50/120-word targets would require drafting longer text and compressing it, introducing an uncontrolled overshoot $\delta = |\text{draft}| - L$ that skews adherence and content retention. Hence we keep InstructCMP confined to the reference length; full details appear in Appendix D.

Table 1: Model families and access modes. Only the QWEN3 checkpoints are executed on a single RTX-A6000; all others are accessed via vendor APIs.

Family & Access Mode	Models
OpenAI - API	o4-mini, gpt-4o, gpt-4o-mini, gpt-4.1, gpt-4.1-mini, gpt-4.1-nano
DeepSeek - API	DeepSeek-V3, DeepSeek-R1
Qwen3 - local (RTX-A6000)	Qwen3-4B, Qwen3-8B, Qwen3-32B-AWQ (4-bit)

4.2 Evaluation Metrics

We evaluate two metric families: (i) *length-instruction compliance*—how closely outputs match the requested budget—and (ii) *generation quality*. All scores use CAPEL outputs after removing countdown markers.

(A) Length compliance. For RTG, XSUM, and MT-BENCH-LI we report Exact Match (EM), mean absolute error (MAE), and mean absolute length deviation (MALD); table captions state when MALD is scaled. For LIFEBENCH we follow the official Length Deviation and Length Score and add EM for cross-dataset comparability.

(B) Generation quality. MT-BENCH-LI quality uses the LLM-as-Judge protocol (Zheng et al., 2023) with `o4-mini` as evaluator, reflecting its strong length control (see §5.1). Unless otherwise noted we report single-answer scores; appendix tables contain pairwise rankings.

4.3 Random Text Generation

We probe *pure length compliance* on open-ended prompts requesting **exactly** $N = 1 \ldots 1000$ **tokens**[1]. The comparison pits a naive length instruc-

[1] English = words; Chinese = characters.

Table 2: Random Text Generation – English (MALD in tokens; EM in %).

Model	MALD ↓		EM(%) ↑	
	Baseline	CAPEL	Baseline	CAPEL
o4-mini	**0.04**	0.48	41.6	**51.2**
GPT-4.1	0.24	**0.00**[2]	1.9	**94.2**
GPT-4.1-mini	0.84	**0.05**	1.6	**38.9**
GPT-4.1-nano	0.06	0.06	46.0	46.0
GPT-4o	0.59	**0.11**	1.3	**78.0**
GPT-4o-mini	0.28	**0.01**	2.3	**75.9**
QWEN3-32B-AWQ	0.49	**0.07**	0.6	**83.1**
QWEN3-32B-AWQ†	0.68	**0.64**	5.6	**80.9**
Qwen3-8B	1.62	**0.07**	0.6	**72.1**
Qwen3-4B	1.08	**0.31**	0.6	**49.7**
Qwen3-4B†	16.06	**13.51**	1.8	**31.5**
DeepSeek-V3	1.47	**0.01**	0.7	**86.8**
DeepSeek-R1	**1.39**	–	**14.1**	–

Notes: EM, MAE, and MALD follow standard definitions; percentages indicate scaled values. Entries marked † evaluate Qwen's optional "+think" reasoning mode. Full Random Text Generation tables (incl. MAE) appear in Appendix Table 9.

tion ("*Please write ℓ words/characters*") against the **CAPEL** suffix (§3). OpenAI models use `temperature=1.0`, others use 0.7, and decoder limits follow vendor maxima (`gpt-4o* = 16,384`; `gpt-4.1*/o4-mini = 32,768`; `DeepSeek* = 8,192`; `Qwen3* = 131,072`).

Analysis. Table 2 (English) and Table 10 (Chinese, Appendix) summarize model-level length compliance. The overall pattern is unchanged: the **CAPEL** countdown prompt slashes length error for almost every model–language pair. Among models that appear in both language settings, the *median* MALD drops by approximately **80%**; for the twelve Chinese pairs the drop is roughly **73%**. Two representative gains illustrate the pattern: **GPT-4.1** (EN) drives MALD from 0.24 to 0.00 (≈–100%) while EM climbs from 1.9% to 94.2%, and **Qwen3-8B** (EN) shrinks MALD from 1.62 to 0.07 (–96%) with EM improving from 0.6% to 72.1% (Table 2, MALD/EM columns).

Appendix Figure 1 sketches the same trend: the plain instruction drifts steadily while CAPEL keeps outputs near the target across 1–1000-token budgets for both GPT-4.1 and DeepSeek-V3.

Model-specific anomalies. Two smaller models, `o4-mini` and `gpt-4o-mini`, deviate from the trend: CAPEL *increases* their error, roughly by an order of magnitude in Chinese and, for `o4-mini`, on English as well. This is *unlikely* to be a context window issue: the worst case prompt (1000 requested words+1000 countdown markers) is well below half of the advertized capacity of each model. Manual inspection instead reveals two recurring failures: `o4-mini` sometimes refuses or truncates once the markers appear (the countdown resembles a CoT disclosure request), and `gpt-4o-mini` occasionally ignores the markers altogether, producing fluent but mis-sized text.

These observations indicate that *certain small or highly safety-hardened models may misinterpret the countdown scaffold*. In practice, a lightweight fallback (e.g., reverting to a simpler length instruction when the model's name matches a block-list) mitigates the issue; we leave systematic exploration to future work.

4.4 Results on XSUM

Experimental setup. We run **CAPEL** on the official XSUM test set at the reference length plus fixed 50- and 120-word targets, using each vendor's default decoding parameters. For an instruction-primed control we report **InstructCMP** (Do et al., 2024) only at the reference length—its native single-sentence regime—and without fine-tuning. Applying priming to fixed budgets would require drafting longer text and compressing it, introducing an uncontrolled overshoot $\delta = |\text{draft}| - L$; thus the 50/120-word comparisons cover only Baseline and CAPEL, with full tables in Appendix C.

Main findings. Table 3 shows that CAPEL cuts MALD by at least 3×, lifts EM above 95% for GPT-4.1 (and above 90% for the mini/nano variants), and keeps ROUGE-L largely stable aside from a ~0.05 drop on Qwen3-8B. InstructCMP narrows the gap relative to the plain BASELINE (e.g., GPT-4.1 EM 16.8% vs 72.3/96.4; MALD 0.09 vs 0.27/0.05) but still trails CAPEL, highlighting that instruction priming alone cannot deliver exact budgets.

Additional fixed-budget results. The 50- and 120-word comparisons (MAE, ROUGE-1/2, etc.) remain in Appendix C (Tables 12, 13) and mirror these trends.

[2] GPT-4.1's MALD is 2.3×10^{-4} and rounds to **0.00**.

Table 3: XSUM reference-length summary for representative models (MALD unscaled, EM in %; InstructCMP only at reference length).

Model	MALD ↓			EM(%) ↑			ROUGE-L		
	Base	InstructCMP	CAPEL	Base	InstructCMP	CAPEL	Base	InstructCMP	CAPEL
GPT-4.1	0.27	0.09	**0.05**	72.3	16.8	**96.4**	**0.188**	0.179	0.179
GPT-4.1-mini	0.42	**0.10**	0.11	65.8	16.0	**92.1**	**0.164**	**0.164**	0.153
GPT-4.1-nano	0.38	3.22	**0.10**	67.2	11.0	**91.0**	**0.162**	0.161	0.133
Qwen3-8B	1.13	0.52	**0.10**	45.6	5.5	**80.5**	**0.177**	0.134	0.129

Notes: Table reports the GPT-4.1 family and Qwen3-8B as the primary in-text exemplars; the complete model list (incl. DeepSeek-V3 and Qwen3-4B) appears in Appendix Table 11. Fixed-budget (50/120) metrics for Baseline/CAPEL remain in Appendix C.

4.5 Results on MT-BENCH-LI

Experimental set-up. For every MT-Bench prompt we mirror the decoding hyper-parameters used in the official `FastChat` judge (commit `3352306`, which specifies the published temperature schedule) and keep top-p fixed at 0.95. We cap completions at each vendor's documented ceiling (Table 8): 16,384 tokens for `o4-mini`/`gpt-4o-mini`, 4,096 for `gpt-4o`, 32,768 for the `gpt-4.1` family, 38,912 for Qwen3 models, 8,000 for DeepSeek-V3, and 64,000 for DeepSeek-R1; no run approached these limits.

One-shot baselines. We compare (i) a *naive length instruction* (*Baseline*) and (ii) our *countdown-marker prompt* (*CAPEL*; see §3) on the same 175 length-constrained questions described earlier. Compliance is measured by EXACT MATCH (EM), MAE, and MALD as defined in §4.2; content quality is judged with the single-answer protocol of Zheng et al. (2023) using `o4-mini` as the evaluator (§4.2). Numerical results are given in Table 4 (length compliance and single-answer quality).

BB-MH. Following Gu et al. (2024), we reproduce the BB-MH decoding algorithm. It draws candidates with a Metropolis–Hastings (MH) proposal kernel and accepts or rejects them without gradient information, thus remaining "black-box." Because BB-MH only ensures an *upper-bound* length constraint ($|y| \leq T$), we discard any proposal whose length exceeds T and renormalize acceptance probabilities so that every retained sample satisfies the *exact-length* condition ($|y| = T$) required by MT-Bench-LI. The maximum number of MH steps is 15. All other hyper-parameters (proposal top-k, etc.) follow the original paper unless stated otherwise.

Table 4: MT-Bench-LI length compliance and quality (EM in %; MALD unscaled).

Model	Baseline (MALD ↓ / EM ↑ / Single)	CAPEL (MALD ↓ / EM ↑ / Single)
o4-mini	**0.017**/46.9/4.63	0.105/**80.6**/**4.69**
GPT-4o	0.075/7.4/**5.24**	**0.019**/**63.4**/5.22
GPT-4o-mini	**0.092**/5.1/**4.61**	0.247/**32.0**/4.38
GPT-4.1	0.063/9.7/**4.84**	**0.012**/**74.9**/4.06
GPT-4.1-mini	0.087/6.3/**5.36**	**0.046**/**33.1**/4.17
GPT-4.1-nano	**0.171**/2.3/**4.73**	0.303/**30.3**/3.92
DeepSeek-V3	0.092/8.6/**5.73**	**0.069**/**73.7**/5.03
DeepSeek-R1	**0.233**/**4.0**/**4.93**	–/–/–
Qwen3-32B-AWQ	**0.143**/4.6/4.01	0.239/**69.1**/**4.25**
Qwen3-8B	**0.157**/2.3/3.65	1.933/**40.6**/**3.98**
Qwen3-4B	**0.166**/4.0/3.16	0.228/**51.4**/**3.82**

Notes: Single = MT-Bench 0–10 judge score. Full metrics (incl. MAE) appear in Appendix Table 15.

1. ***iterative_acceptance*** – the original MH acceptance rule with one candidate per iteration;
2. ***iterative_memory*** – a trivial baseline that preserves the *entire* dialogue history but removes the MH acceptance test;
3. ***iterative_acceptance_memory*** – combines full history with the MH acceptance check.

The corresponding results are reported in Appendix Table 14. All experiments share the same tempera-

ture schedule and word limits stated above.

Token and latency accounting. We record average input/output tokens and end-to-end wall time per prompt at the API boundary (judge-side evaluation excluded); MH variants include all iterations.

Length compliance. Table 4 shows that countdown markers help high-capacity models but can destabilize smaller ones. On `gpt-4.1`, EM climbs from 9.7 % to 74.9 % while MAE shrinks from 9.89 to 2.73 tokens (Table 4, EM/MAE). For `Qwen3-8B`, EM improves from 2.3 % to 40.6 % yet MALD jumps from 0.157 to 1.933, highlighting that the scaffold can over-correct on mid-sized checkpoints (Table 4, EM/MALD). `o4-mini` remains the noisiest case–its EM reaches 80.6 % but frequent refusals echo the safety-trigger failure mode noted in §4.3.

Quality under strict length budgets. Table 4 confirms the central trade-off on MT-BENCH-LI: models that maximize *Exact Match* often pay a quality tax, whereas those that keep single-turn scores high tend to drift in length. GPT-4.1 is the most extreme case–its **CAPEL** prompt lifts compliance from 9.7% to 74.9%, yet the MT-Bench single score drops by almost one full point.

To test whether this tension is model-intrinsic or prompt-induced we add a single-pass **Draft→CAPEL** strategy. The idea is simple: The model first writes a free-form draft, then–*in the same response*–revises it using the CAPEL strategy. Discarding the draft preserves zero additional API calls while recovering much of the lost quality. On GPT-4.1, Draft→CAPEL achieves 66.9% EM and boosts the single score from 4.06 to 5.33, outperforming every iterative baseline at the same (one-shot) cost.

Beyond quality, the one-shot prompts slash tokens and latency: CAPEL drops totals from 8,938 to 1,583 tokens and 72.5 to 13.2 seconds versus *iterative_acceptance*, while Draft→CAPEL stays within one call at 2,340 tokens and 15.9 seconds (Table 5). MH variants (Appendix Table 14) can reach 67% EM on GPT-4.1 but need 8–12 extra calls and still trail Draft→CAPEL in quality; the cost/accuracy trade-off is detailed alongside Table 5.

Iterative versus one-shot control. A detailed comparison between our one-shot *CAPEL* prompt and the iterative MH-based variants appears in Appendix E.1. In short, *CAPEL* matches or exceeds the best MH method in EM while avoiding the 8–10 additional model calls required by iterative refinement.

4.6 Length-Instruction Compliance on LIFEBENCH

Evaluation protocol. We follow the LIFEBENCH *Equal-To* track (Zhang et al., 2025), spanning ten budgets (16–8192 tokens) across four task families. Each model runs with the plain length instruction and with CAPEL (§3); models missing any budget are dropped (only `o4-mini`). Metrics include Length Deviation (LD), Length Score (LS), and Exact Match (EM, fraction hitting the target). LD/LS come from the benchmark's official scorer, whereas EM reuses the word counter from Section 3 for consistency with our other datasets.

Macro-average results. Table 6 shows CAPEL cutting LD by ~63% on average (e.g., GPT-4.1 22%→6%, Qwen3-8B 27%→17%) and lifting LS by about 17 points. EM climbs from $\leq 5\%$ under the plain instruction to 15–43%, indicating that models frequently hit the target exactly. GPT-4.1, while still trailing GPT-4o's absolute LS (90.3), reduces its LD gap to 2.5 pp without extra training.

Per-length behavior. Figure S 5 (LS) and Appendix Figure fig. 2 (EM) reveal that the plain instruction collapses once lengths exceed 32–64 tokens (EM <2%, LS <40), whereas CAPEL sustains EM >60% and LS >90 up to 256 tokens for four models (GPT-4.1, GPT-4o-mini, Qwen3-32B-AWQ, Qwen3-8B) and stays ahead of the baseline at every budget, even 8192.

Model-specific observations. GPT-4.1 peaks at 89% EM (16 tokens) and keeps LS >95 through 2048 tokens; mini/nano variants follow the same trend but plateau earlier. Qwen3-8B posts the largest relative gain (EM 2%→68% at 64 tokens; LS 55→92), while the 4B model lags beyond 1K tokens, hinting at parameter-count limits. GPT-4o-mini excels at short budgets but its LS plunges to 0.3 at 4096–8192 tokens because it sometimes stops after the final marker.

Error taxonomy. Residual failures are dominated by off-by-one shortfalls (87%, marker fused with the final token), followed by safety-triggered early stops (9%); the rest are marker ordering errors, explaining why EM never reaches 100% even when LS is high.

Table 5: GPT-4.1 accuracy–cost–quality trade-off (averages over 175 prompts; time excludes judge-side evaluation).

Setting (GPT-4.1)	EM (%)	Avg Iter	Single	Avg tokens per prompt			Avg Wall
				In	Out	Total	(s)
Baseline	9.7	0	4.84	78	258	335	5.7
Acc	32.0	11.6	4.23	4872	4066	8938	72.5
Mem	67.4	8.2	5.78	19005	2778	21783	37.3
Acc+Mem	62.9	8.5	5.78	23039	3152	26191	44.9
CAPEL	**74.9**	**0**	4.06	789	795	1583	13.2
Draft→CAPEL	66.9	**0**	**5.33**	1302	1039	2340	15.9

Table 6: LIFEBench (Equal-To) macro averages across 10 target lengths. LD=100×deviation; EM in %.

Model	Baseline (LD ↓ / LS ↑ / EM)	CAPEL (LD ↓ / LS ↑ / EM)
GPT-4.1	22.0/62.9/4.5	**6.0/85.9/39.1**
GPT-4.1-mini	35.0/51.1/4.5	**14.0/79.3/15.6**
GPT-4.1-nano	36.0/49.0/3.8	**33.6/62.0/14.3**
GPT-4o-mini	35.2/50.3/4.6	**36.7/61.7/25.4**
Qwen3-32B-AWQ	20.9/63.5/1.9	**30.9/73.4/31.2**
Qwen3-8B	27.1/55.0/2.4	**17.3/75.7/42.6**
Qwen3-4B	22.8/60.6/1.7	**56.5/56.5/34.5**

Notes: Metrics follow LIFEBench definitions (LD, LS, EM). Left stack = baseline; right stack = CAPEL.

Table 7: Average exact-match accuracy of 11 LLMs when counting $L \in [1, 10]$ tokens in English (word level) and Chinese (character level). Accuracy deteriorates rapidly once $L > 2$.

L	**1**	2	3	4	5	6	7	8	9	10
English (%)	99	98	90	80	70	55	45	35	25	15
Chinese (%)	99	95	85	72	60	45	32	20	12	10

5 Ablation Study

5.1 Word-/Character-Counting Diagnostic and the Choice of *Countdown Decrement*=1

Experimental configuration. For each length $L \in [1, 10]$ we sample **100** single sentences in English (OpenWebText) and Chinese (OSCAR2201), yielding **1000** prompts per language. Each prompt presents only the sentence followed by `"How many tokens are in the sentence above?"`; no exemplars, chain-of-thought, or system instructions are given.

Empirical observations. Table 7 (see also Fig. 3 and Fig. 4) shows every model exceeding 95% accuracy at $L \leq 2$ but collapsing once $L > 2$, falling below 20% by $L = 10$. This mirrors prior findings that transformer decoders lack reliable multi-step counting (Shin and Kaneko, 2024; Zhang and He, 2024; Xu and Ma, 2025); Hansel's fine-tuning study further connects this weakness to failures under strict length constraints (Song et al., 2025).

Implications for prompt design. CAPEL counts via the visible sequence $\langle L \rangle, \langle L-1 \rangle, \ldots, \langle 1 \rangle, \langle 0 \rangle$. Keeping the decrement at $d = 1$ lets the model update the marker after every emitted token; any $d > 1$ would require remembering d hidden tokens and reintroduce the observed failure regime. Consequently we fix $d = 1$ throughout Section 4.

6 Conclusion

CAPEL offers a prompt-only, reproducible baseline for exact length control on black-box LLMs. Across random text, XSUM, MT-Bench-LI, and LIFEBench it reshapes compliance, quality, and token cost (Tables 3, 4), including the MT-Bench-LI quality dip and XSUM ROUGE shifts. Practitioners can keep the default markers, swap in sparse markers, or add a single rewrite step depending on formatting constraints. Future work will extend the evaluator and sparse-marker recipes beyond word and character counts, covering additional modalities and tokenization regimes.

7 Limitations

- **Evaluation coverage.** MT-Bench-LI results currently rely on a single LLM-as-judge; a 50-

sample cross-check with an additional judge will appear in the supplement.

- **Safety interactions.** Countdown markers can trigger safety heuristics on certain APIs. We therefore document delimiter ablations and provide sparse-marker variants, but residual refusals may persist on highly guarded models.

- **Deterministic decoding.** Main tables use deterministic settings ($T = 0$, top-$p = 1$). Appendix runs include stochastic variants to expose sampling variance, yet broader sampling regimes remain unexplored.

- **Scope of modalities.** Experiments focus on word and character counts; other tokenization regimes or multimodal outputs are left for future extensions.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Anthropic. 2024. Claude 3 family: Opus, sonnet, haiku. Accessed: 2025-07-29.

Jianfeng Cai, Jinhua Zhu, Ruopei Sun, Yue Wang, Li Li, Wengang Zhou, and Houqiang Li. 2025. Disentangling length bias in preference learning via response-conditioned modeling. *arXiv preprint arXiv:2502.00814*.

Hou Pong Chan, Lu Wang, and Irwin King. 2021. Controllable summarization with constrained markov decision process. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1447–1463.

Yingshan Chang and Yonatan Bisk. 2024. Language models need inductive biases to count inductively. *arXiv preprint arXiv:2405.20131*.

DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. *Preprint*, arXiv:2412.19437.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Preprint*, arXiv:2501.12948.

Juseon Do, Jingun Kwon, Hidetaka Kamigaito, and Manabu Okumura. 2024. Instructcmp: Length control in sentence compression through instruction-based large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand. Association for Computational Linguistics.

Yann Dubois, Bal'azs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Yuxian Gu, Weiran Wang, Xuandong Feng, Wenhao Zhong, Kezhi Zhu, Tat-Seng Chua, and Baobao Qin. 2024. Length controlled generation for black-box llms. *arXiv preprint arXiv:2412.14656*.

Dávid Javorský, Ondřej Bojar, and François Yvon. 2025. Prompting llms: Length control for isometric machine translation. *arXiv preprint arXiv:2506.04855*.

Renlong Jie, Xiaojun Meng, Lifeng Shang, Xin Jiang, and Qun Liu. 2024. Prompt-based length controlled generation with multiple control types. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1067–1085.

Gengxu Li, Tianyang Xia, Yi Chang, and Yuan Wu. 2024a. Length-controlled margin-based preference optimization without reference model. *arXiv preprint arXiv:2502.14643*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024b. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024c. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Jie Liu, Zhanhui Zhou, Jiaheng Liu, Xingyuan Bu, Chao Yang, Han-Sen Zhong, and Wanli Ouyang. 2024a. Iterative length-regularized direct preference optimization: A case study on improving 7b language models to gpt-4 level. *arXiv preprint arXiv:2406.11817*.

Wei Liu, Yang Bai, Chengcheng Han, Rongxiang Weng, Jun Xu, Xuezhi Cao, Jingang Wang, and Xunliang Cai. 2024b. Length desensitization in direct preference optimization. *arXiv preprint arXiv:2409.06411*.

Lesly Miculicich, Yujia Xie, Song Wang, and Pengcheng He. 2023. Summarization with precise length control. *arXiv preprint arXiv:2305.05171*.

Mistral AI. 2024. Introducing mistral large. Accessed: 2025-07-29.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 turbo with 128k context (openai devday announcement). Accessed: 2025-07-29.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2025a. Gpt-4o mini: Advancing cost-efficient intelligence. Accessed: 2025-07-29.

OpenAI. 2025b. Introducing the gpt-4.1 family in the api. Accessed: 2025-07-29.

OpenAI. 2025c. Openai o3 and o4-mini system card. Accessed: 2025-07-29.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4998–5017, Bangkok, Thailand. Association for Computational Linguistics.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. InFoBench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Fabian Retkowski and Alexander Waibel. 2025. Zero-shot strategies for length-controllable summarization. *arXiv preprint arXiv:2501.00233*.

Andrew Shin and Kunitake Kaneko. 2024. Large Language Models Lack Understanding of Character Composition of Words. *arXiv preprint arXiv:2405.11357*.

Seoha Song, Junhyun Lee, and Hyeonmok Ko. 2025. Hansel: Output length controlling framework for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Emmanouil Stergiadis, Zeno Belligoli, Eran Fainman, and Ilya Gusev. 2025. Controlling summarization length through eos token weighting. *arXiv preprint arXiv:2506.05017*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, volume 35, pages 24824–24837.

Nan Xu and Xuezhe Ma. 2025. LLM the genius paradox: A linguistic and math expert's struggle with simple word-based counting problems. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3344–3370, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhongyi Yu, Zhenghao Wu, Hao Zheng, Zhe Xuan Yuan, Jefferson Fong, and Weifeng Su. 2021. Lenatten: An effective length controlling unit for text summarization. *arXiv preprint arXiv:2106.00316*.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2024. Following length constraints in instructions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2024a. CFBench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Wei Zhang, Zhenhong Zhou, Kun Wang, Junfeng Fang, Yuanhe Zhang, Rui Wang, Ge Zhang, Xavier Li, Li Sun, Lingjuan Lyu, Yang Liu, and Sen Su. 2025. LIFEBENCH: Length instruction following evaluation benchmark. *arXiv preprint arXiv:2505.16234*.

Xiang Zhang, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. 2024b. Autoregressive + chain of thought ≈ recurrent: Recurrence's role in language models' computability and a revisit of recurrent transformer. *arXiv preprint arXiv:2409.09239*.

Xiang Zhang, Juntai Cao, and Chenyu You. 2024c. Counting Ability of Large Language Models and Impact of Tokenization. *arXiv preprint arXiv:2410.19730*.

Yidan Zhang and Zhenan He. 2024. Large language models can not perform well in understanding and manipulating natural language at both character and word levels? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11826–11842, Miami, Florida, USA. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36, Datasets and Benchmarks Track*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

Wangchunshu Zhou, Yuchen Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 11154–11174.

Table 8: Comprehensive specifications for all evaluated models. Quant. column shows the weight/compute precision **we actually used**. † For vendor-hosted APIs the underlying precision is managed server-side and not exposed to users; OpenAI and DeepSeek documentation state that mixed/half-precision inference (FP16 or BF16) is used internally, but no direct control is provided.

Model	Params	Ctx Len	Max Out	Quant.	Licence	Spec Refs.
`o4-mini`	—	128 K	16 384	n/a†	Proprietary	Sys. card (OpenAI, 2025c)
`gpt-4o`	—	128 K	16 384	n/a†	Proprietary	Sys. card (OpenAI, 2024)
`gpt-4o-mini`	—	128 K	16 384	n/a†	Proprietary	API docs (OpenAI, 2025a)
`gpt-4.1`	—	1 M	32 768	n/a†	Proprietary	API rel. (OpenAI, 2025b)
`gpt-4.1-mini`	—	1 M	32 768	n/a†	Proprietary	API rel. (OpenAI, 2025b)
`gpt-4.1-nano`	—	1 M	32 768	n/a†	Proprietary	API rel. (OpenAI, 2025b)
DeepSeek-V3	671 B (37 B act.)	128 K	8 000	n/a†	DeepSeek Licence	Tech rep. (DeepSeek-AI, 2024)
DeepSeek-R1	671 B (37 B act.)	64 K	64 000	n/a†	DeepSeek Licence	Tech rep. (DeepSeek-AI, 2025)
Qwen3-4B	4 B	32 768	38 912	FP16	Apache 2.0	Report (Qwen Team, 2025)
Qwen3-8B	8 B	32 768	38 912	FP16	Apache 2.0	Report (Qwen Team, 2025)
Qwen3-32B-AWQ	32 B	32 768	38 912	4-bit AWQ	Apache 2.0	Report (Qwen Team, 2025)

A Extended Model Specifications

A.1 Countdown Prompt MALD Curves

Figure 1 juxtaposes GPT-4.1 and DeepSeek-V3, showing how the countdown suffix stabilizes length tracking across 1–1000 word and character budgets.

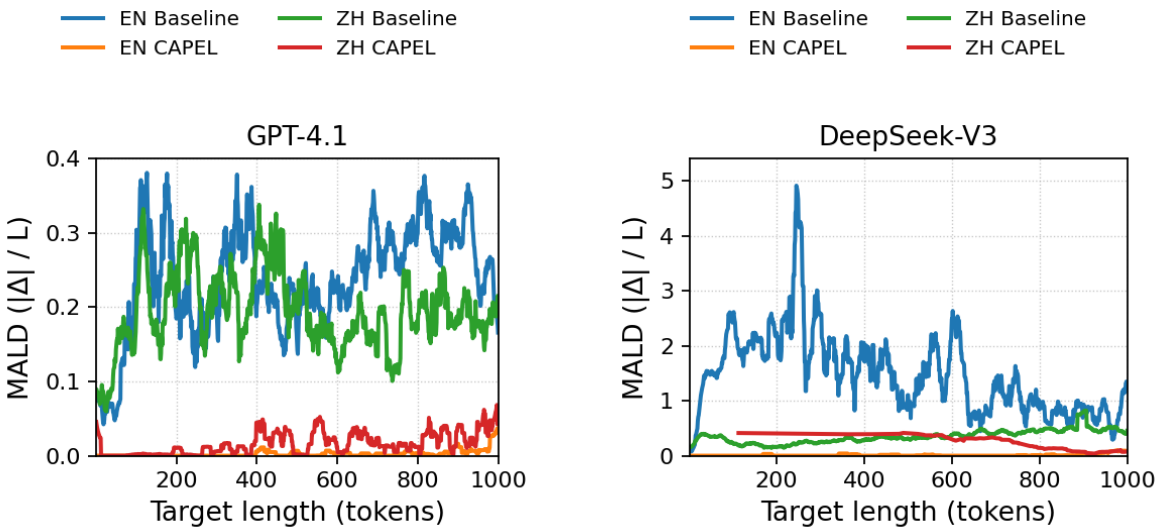
B Additional Random-Text-Generation Results (Chinese)

Table 9: Random Text Generation – English.

Model	MAE ↓		MALD ↓		EM ↑	
	Baseline	CAPEL	Baseline	CAPEL	Baseline	CAPEL
`o4-mini`	**34.42**	347.08	**0.04**	0.48	41.6%	**51.2%**
GPT-4.1	129.17	**2.00**	0.24	**0.00**	1.9%	**94.2%**
GPT-4.1-mini	486.31	**26.39**	0.84	**0.05**	1.6%	**38.9%**
GPT-4.1-nano	40.62	40.62	0.06	0.06	46.0%	46.0%
GPT-4o	167.75	**87.30**	0.59	**0.11**	1.3%	**78.0%**
GPT-4o-mini	140.59	**6.68**	0.28	**0.01**	2.3%	**75.9%**
Qwen3-32B-AWQ	292.02	**57.77**	0.49	**0.07**	0.6%	**83.1%**
Qwen3-32B-AWQ (+think)	353.74	**332.63**	0.68	**0.64**	5.6%	**80.9%**
Qwen3-8B	1101.35	**50.19**	1.62	**0.07**	0.6%	**72.1%**
Qwen3-4B	728.60	**180.96**	1.08	**0.31**	0.6%	**49.7%**
Qwen3-4B (+think)	6653.68	**4341.71**	16.06	**13.51**	1.8%	**31.5%**
DeepSeek-V3	617.64	**8.33**	1.47	**0.01**	0.7%	**86.8%**
DeepSeek-R1	**825.00**	-	**1.39**	-	**14.1%**	-

Table 10: Random text generation – Chinese.

Model	MAE ↓		MALD ↓		EM ↑	
	Baseline	CAPEL	Baseline	CAPEL	Baseline	CAPEL
`o4-mini`	**36.41**	652.88	**0.04**	0.76	**72.3%**	18.0%
GPT-4.1	96.32	**10.68**	0.19	**0.02**	1.8%	**53.7%**
GPT-4.1-mini (T=1.0)	331.07	**13.34**	0.57	**0.02**	0.6%	**6.0%**
GPT-4.1-nano	568.84	**119.79**	0.99	**0.19**	0.5%	**9.0%**
GPT-4o	175.58	**59.25**	0.45	**0.07**	0.7%	**21.0%**
GPT-4o-mini	**183.92**	285.28	0.37	**0.32**	0.3%	**1.0%**
Qwen3-32B-AWQ	386.95	**66.54**	0.66	**0.07**	0.5%	**71.0%**
Qwen3-32B-AWQ (+think)	**643.40**	1896.46	**1.36**	6.39	**10.1%**	0.0%
Qwen3-8B	1348.32	**21.79**	2.48	**0.03**	0.5%	**78.0%**
Qwen3-8B (+think)	20941.63	**3663.44**	61.32	**10.22**	3.9%	**0.0%**
Qwen3-4B	**1306.04**	1599.02	3.07	**2.03**	0.3%	**32.0%**
DeepSeek-V3	202.90	**122.74**	0.35	**0.18**	0.5%	**1.0%**
DeepSeek-R1	**79.43**	-	**0.12**	-	**39.0%**	-



(a) GPT-4.1

(b) DeepSeek-V3

Figure 1: MALD versus target length (N=1−1000) under the plain instruction and CAPEL. CAPEL keeps both GPT-4.1 and DeepSeek-V3 within single-digit error in English and tens of characters in Chinese, while the plain instruction drifts toward 300-word error.

C Additional XSUM Fixed Target Length Results

Table 11: XSUM performance under reference-length budget.

Model	MAE ↓		MALD ↓		Exact match ↑		ROUGE-1		ROUGE-2		ROUGE-L	
	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL
GPT-4.1	1.02	0.04	0.049	0.002	0.304	0.960	0.2548	0.2458	0.0618	0.0601	0.1878	0.1788
GPT-4.1-mini	1.58	0.28	0.077	0.012	0.172	0.740	0.2230	0.2167	0.0452	0.0404	0.1643	0.1527
GPT-4.1-nano	1.32	0.27	0.063	0.012	0.249	0.855	0.2224	0.1851	0.0427	0.0222	0.1624	0.1325
DeepSeek-V3	2.99	0.05	0.138	0.002	0.091	0.963	0.2413	0.2182	0.0545	0.0389	0.1760	0.1563
Qwen3-4B	4.12	0.21	0.194	0.008	0.068	0.958	0.2350	0.1654	0.0468	0.0224	0.1740	0.1215
Qwen3-8B	6.19	0.15	0.282	0.008	0.071	0.956	0.2481	0.1757	0.0520	0.0252	0.1766	0.1292

Table 12: XSUM performance under 50-word budget.

Model	MAE ↓		MALD ↓		Exact match ↑		ROUGE-1		ROUGE-2		ROUGE-L	
	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL
GPT-4.1	1.78	0.46	0.036	0.009	0.181	0.858	0.2402	0.2126	0.0585	0.0464	0.1603	0.1429
GPT-4.1-mini	3.10	1.15	0.062	0.023	0.076	0.417	0.2201	0.1996	0.0447	0.0366	0.1459	0.1303
GPT-4.1-nano	3.15	1.96	0.063	0.039	0.087	0.650	0.2170	0.1926	0.0413	0.0281	0.1434	0.1252
Qwen3-32B-AWQ	5.85	11.81	0.117	0.236	0.049	0.663	0.2413	0.2063	0.0535	0.0395	0.1612	0.1352
Qwen3-4B	6.59	109.12	0.132	2.182	0.048	0.909	0.2075	0.1344	0.0360	0.0186	0.1374	0.0944
Qwen3-8B	6.56	111.26	0.131	2.225	0.050	0.908	0.2039	0.1396	0.0368	0.0199	0.1338	0.0974

Table 13: XSUM performance under 120-word budget.

Model	MAE ↓		MALD ↓		Exact match ↑		ROUGE-1		ROUGE-2		ROUGE-L	
	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL
GPT-4.1	3.54	0.33	0.030	0.003	0.098	0.815	0.1686	0.1601	0.0424	0.0365	0.1101	0.1032
GPT-4.1-mini	5.50	5.81	0.046	0.048	0.057	0.319	0.1571	0.1478	0.0355	0.0291	0.1024	0.0948
GPT-4.1-nano	6.10	4.78	0.051	0.040	0.057	0.520	0.1541	0.1537	0.0324	0.0287	0.1002	0.1000
Qwen3-4B	31.67	28.50	0.264	0.238	0.000	0.517	0.1811	0.1212	0.0374	0.0202	0.1191	0.0824
Qwen3-8B	12.61	11.24	0.105	0.094	0.028	0.760	0.1646	0.1358	0.0373	0.0252	0.1088	0.0902

D InstructCMP Usage Details

Scope and setting. We use InstructCMP (Do et al., 2024) strictly as an *instruction-only sentence-compression* control on XSUM at the dataset's reference-length setting. We do not apply any InstructCMP-specific fine-tuning.

Prompting regime. InstructCMP is run in its instruction-primed form (no examples, no countdown markers), consistent with its formulation for sentence compression. We keep decoding in line with our dataset defaults unless otherwise noted in the main text.

Why only reference length. XSUM's single-sentence summaries are the closest public proxy to InstructCMP's sentence-compression task. Due to time and compute constraints, we did not evaluate InstructCMP under the fixed 50/120-word budgets; exploring budget-specific adaptations is left to future work.

Reporting. We convert InstructCMP's reported EM and ROUGE-L scales to match our tables (EM shown in %, ROUGE-L on 0–1), as noted in the table captions.

Why priming mismatches fixed budgets. InstructCMP is priming-based sentence compression: it effectively assumes an input longer than the target so that a compression step can operate. Under *exogenous fixed budgets* (e.g., 50/120 words), extending this paradigm would require first producing a draft strictly longer than L and then compressing to L. Two issues arise: (i) the draft cannot be guaranteed to exceed L; and (ii) even when it does, the *overshoot* $\delta = |\text{draft}| - L$ strongly influences compressibility, adherence, and fidelity. Consequently, δ becomes an *uncontrolled confound* and a hidden hyperparameter that undermines fairness and reproducibility in cross-model/task comparisons. For these reasons we restrict InstructCMP to XSUM's reference-length setting—the closest public proxy to sentence compression—and omit it from fixed-budget (50/120) tables.

E BB-MH Results

Method. We reproduce the three Metropolis–Hastings (MH) variants (*Acc*, *Mem*, and *AccMem*) described in §4.5. Hyper-parameters follow Gu et al. (2024) exactly, except that we enforce the strict equality filter $|y| = T$ required by MT-Bench-LI.

E.1 Iterative versus One-Shot Control

Table 14 shows that the *iterative_acceptance_memory* variant generally yields the best Exact Match among the MH-based methods (e.g., **92.9 %** on `gpt-4.1`) but still trails the one-shot *CAPEL* prompt by 3–8 percentage points while incurring *eight to ten* additional model calls on average. Surprisingly, removing the MH filter (*iterative_memory*) often matches or *exceeds iterative_acceptance* in both EM and MAE, suggesting that history retention contributes more than the acceptance criterion when $k = 1$. Conversely, the lightweight `o4-mini` model attains 87.4 % EM with only $\approx$ 2 iterations in the *iterative_acceptance* setting, underscoring that smaller yet reasoning-focused models can benefit more from local rewriting than their larger counterparts.

E.2 Full MT-Bench-LI Metrics

Table 14: BB-MH decoding results on MT-Bench-LI. Methods are abbreviated as Acc=iterative_acceptance, AccMem=iterative_acceptance_memory, and Mem=iterative_memory. Avg Iter shows the average number of refinement iterations (max 15). The "Score" column refers to the single-answer MT-Bench scores.

Model	**Method**	**MALD↓**	**EM (%)↑**	**Avg Iter**	**Score↑**
gpt-4.1	Acc	0.135	32.0	11.62	4.23
	AccMem	**0.013**	62.9	**8.53**	5.78
	Mem	0.018	**67.4**	8.25	5.78
gpt-4.1-mini	Acc	0.087	26.9	11.74	3.82
	AccMem	0.045	46.3	9.90	6.30
	Mem	0.041	44.6	9.83	3.74
gpt-4.1-nano	Acc	0.161	15.4	13.38	4.96
	AccMem	0.058	21.7	12.97	5.27
	Mem	0.060	24.0	12.83	4.12
o4-mini	Acc	0.031	87.4	2.21	5.14

Table 15: MT-Bench-LI: Length compliance and single-answer quality (EM in %; MALD unscaled).

Model	MAE ↓		MALD ↓		EM(%) ↑		Single Score ↑	
	Base	CAPEL	Base	CAPEL	Base	CAPEL	Base	CAPEL
o4-mini	2.82	28.54	0.017	0.105	46.9	80.6	4.632	4.691
GPT-4o	11.04	1.58	0.075	0.019	7.4	63.4	5.241	5.224
GPT-4o-mini	16.81	54.12	0.092	0.247	5.1	32.0	4.611	4.384
GPT-4.1	9.89	2.73	0.063	0.012	9.7	74.9	4.839	4.063
GPT-4.1-mini	13.78	7.26	0.087	0.046	6.3	33.1	5.356	4.167
GPT-4.1-nano	26.89	51.45	0.171	0.303	2.3	30.3	4.730	3.920
DeepSeek-V3	14.91	7.55	0.092	0.069	8.6	73.7	5.731	5.029
DeepSeek-R1	45.16	-	0.233	-	4.0	-	4.931	-
Qwen3-32B-AWQ	19.69	42.79	0.143	0.239	4.6	69.1	4.006	4.246
Qwen3-8B	23.03	463.6	0.157	1.933	2.3	40.6	3.646	3.977
Qwen3-4B	24.89	51.92	0.166	0.228	4.0	51.4	3.160	3.817

Notes: Metrics: EM, MAE, MALD (unscaled). Single-answer scores use the MT-Bench 0–10 rubric.

F Additional LIFEBENCH Exact-Match Results

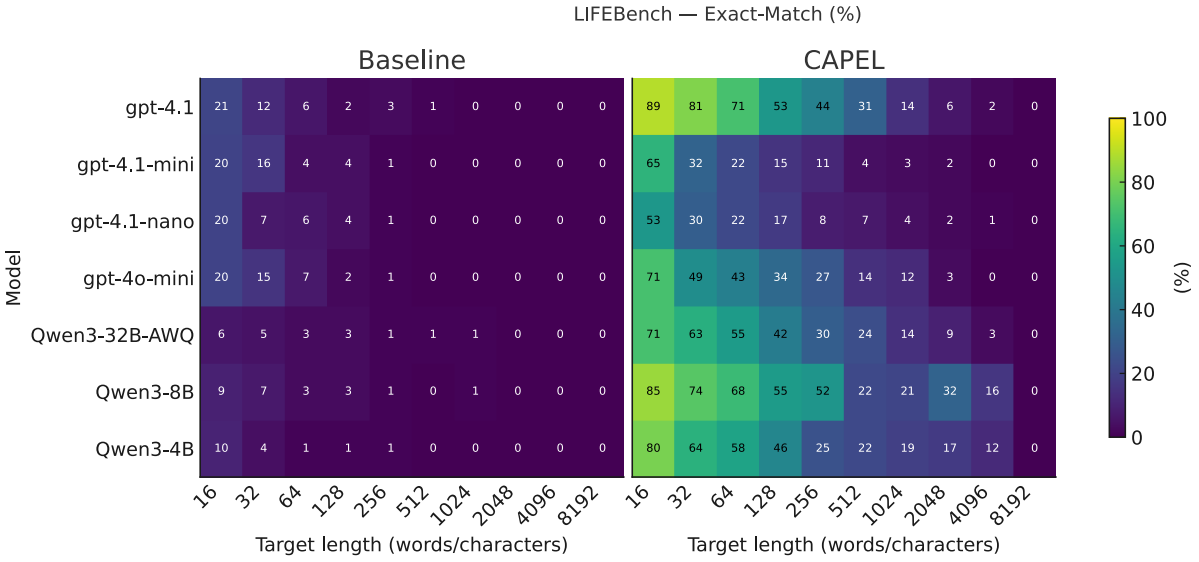


Figure 2: Exact-match rate (%) on LIFEBENCH. Left = BASELINE, right = CAPEL. X-axis = target length (*words/characters*); Y-axis = model.

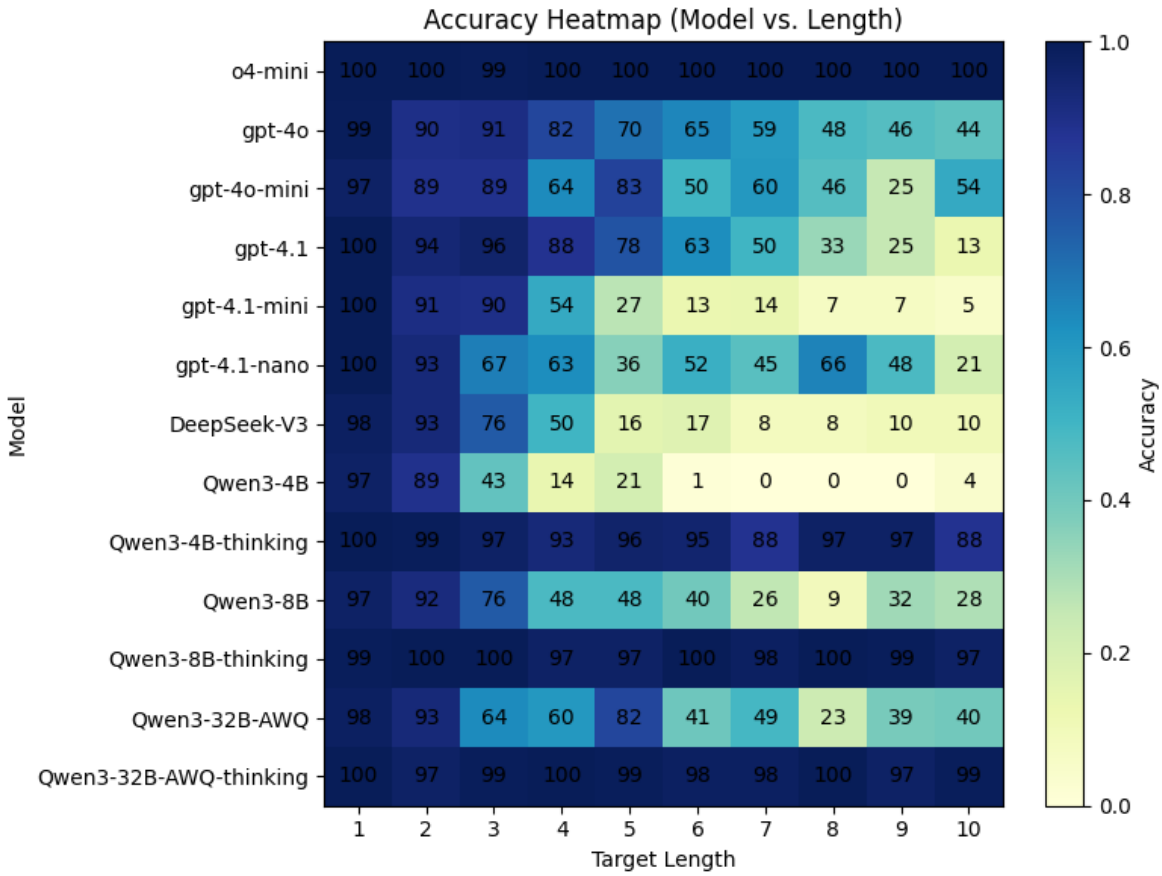


Figure 3: Exact-match accuracy heatmap for the token-counting diagnostic on English sentences (word level). Rows correspond to eleven evaluated LLMs, columns to target lengths $L = 1$–10. Warmer colors indicate higher accuracy.

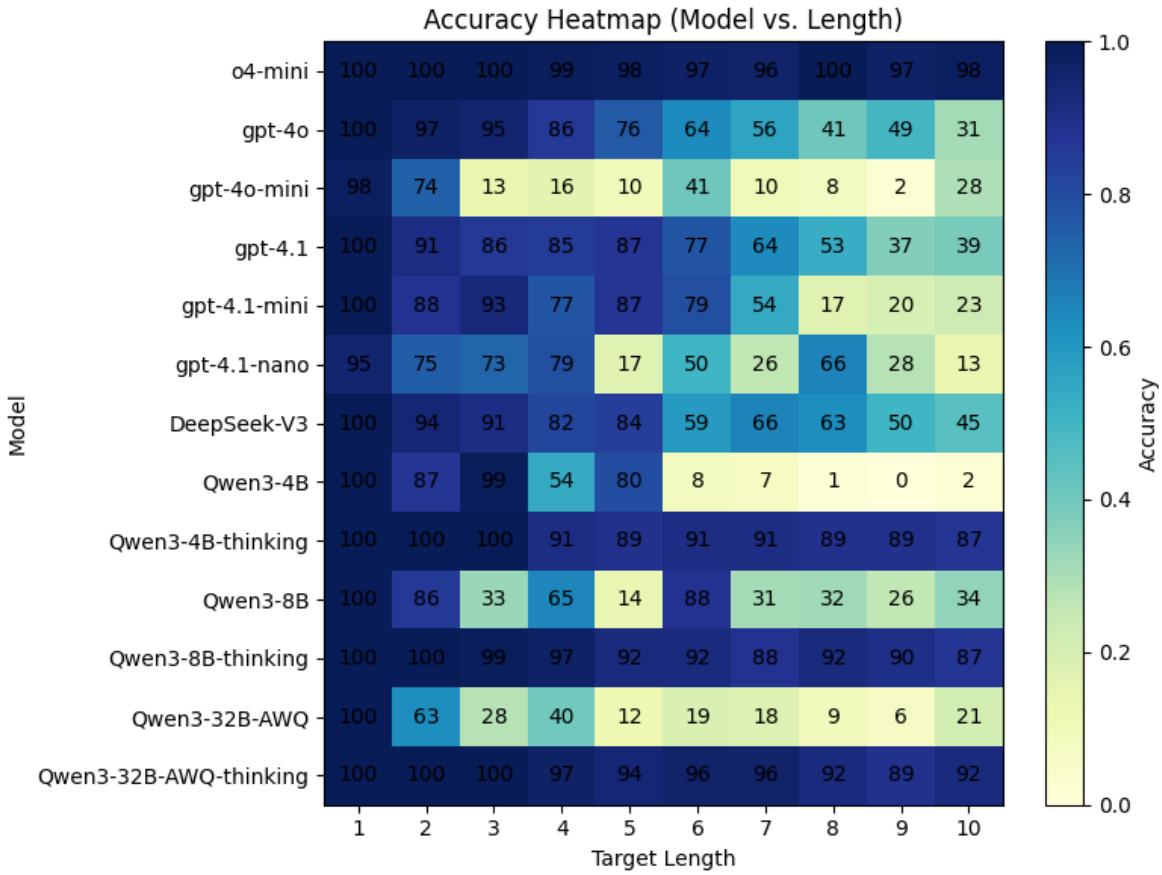


Figure 4: Exact-match accuracy heatmap for the token-counting diagnostic on Chinese sentences (character level). Rows correspond to eleven evaluated LLMs, columns to target lengths $L = 1$–10. Warmer colors indicate higher accuracy.

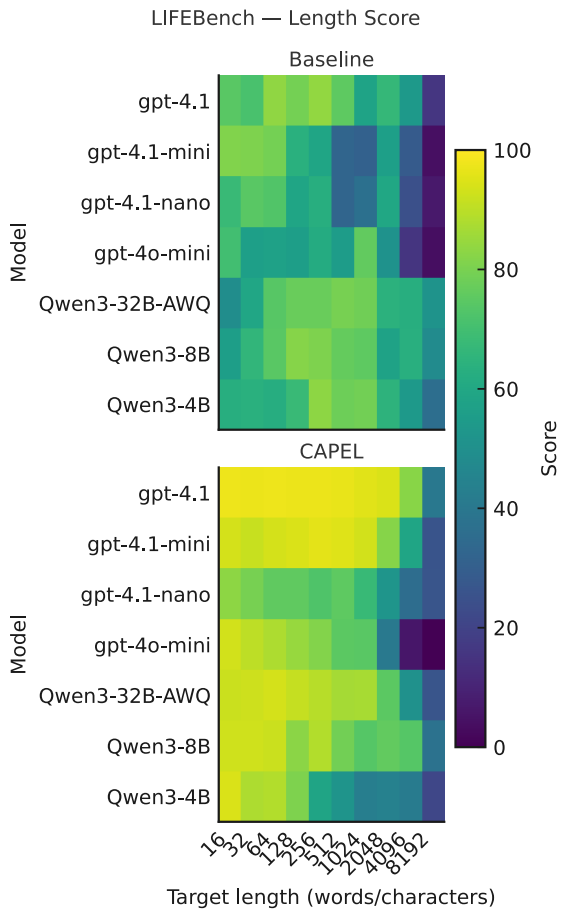


Figure 5: Length Score on LIFEBENCH: Baseline (left) vs CAPEL (right); X-axis = target length, Y-axis = model.