From Sub-Ability Diagnosis to Human-Aligned Generation: Bridging the Gap for Text Length Control via MARKERGEN

Peiwen Yuan[1*], **Chuyi Tan**[1*], **Shaoxiong Feng**[2], **Yiwei Li**[1], **Xinglin Wang**[1]
Yueqi Zhang[1], **Jiayi Shi**[1], **Boyuan Pan**[2], **Yao Hu**[2], **Kan Li**[1†]
[1]School of Computer Science and Technology, Beijing Institute of Technology
[2]Xiaohongshu Inc
{peiwenyuan,tanchuyi,liyiwei,wangxinglin,zhangyq,shijiayi,likan}@bit.edu.cn
{shaoxiongfeng2023,whd.thu}@gmail.com {panboyuan,xiahou}@xiaohongshu.com

Abstract

Despite the rapid progress of large language models (LLMs), their length-controllable text generation (LCTG) ability remains below expectations, posing a major limitation for practical applications. Existing methods mainly focus on end-to-end training to reinforce adherence to length constraints. However, the lack of decomposition and targeted enhancement of LCTG sub-abilities restricts further progress. To bridge this gap, we conduct a bottom-up decomposition of LCTG sub-abilities with human patterns as reference and perform a detailed error analysis. On this basis, we propose MARKERGEN, a simple-yet-effective plug-and-play approach that: (1) mitigates LLM fundamental deficiencies via external tool integration; (2) conducts explicit length modeling with dynamically inserted markers; (3) employs a three-stage generation scheme to better align length constraints while maintaining content quality. Comprehensive experiments demonstrate that MARKERGEN significantly improves LCTG across various settings, exhibiting outstanding effectiveness and generalizability.[1].

1 Introduction

As a fundamental attribute of text generation, ensuring controllability over text length is of great importance (Liang et al., 2024). Different text types (e.g., summary, story), user needs (e.g., preference for detailed or concise writing), and external requirements (e.g., social media character limits) shape varied length constraints, which are widely present in real-world scenarios (Zhang et al., 2023a). With the rapid development of LLMs, their expanding range of applications has made length-controllable text generation (LCTG) even more crucial in current era (Foster et al., 2024; Gu et al., 2024b).

However, the ongoing enhancements in LLM capabilities have yet to deliver the expected performance in LCTG while ensuring semantic integrity (Foster et al., 2024; Wang et al., 2024; Song et al., 2024). Yuan et al. (2024) reports that even advanced LLMs (e.g., GPT-4 Turbo (OpenAI, 2023)) violate the given length constraints almost 50% of the time. To address this, training-based methods (Park et al., 2024; Yuan et al., 2024; Jie et al., 2023; Li et al., 2024b) have been studied to reinforce LLMs' adherence to length constraints, yet they face two key challenges: (1) **Limited generalization**: Since text types are diverse and length constraints vary widely (e.g., ranging from an exact 500 words to coarse intervals like 500-600 words or below 500 words), training-based methods often fail to generalize effectively across different settings, as demonstrated in Appendix E.1. (2) **Inferior controllability**: These methods strengthen LCTG by enforcing implicit length modeling during generation in a top-down manner via training, lacking the decomposition and targeted enhancement of LCTG sub-capabilities, thereby limiting their progress (Retkowski and Waibel, 2024).

To fill this gap, we take humans as a reference and conduct a bottom-up decomposition of sub-capabilities for LCTG. When writing a 500-word story, humans typically begin by planning the content and word allocation for each section. During writing, they continuously track the word count and compose the text in alignment with the plan. This process progressively tests four key abilities: (1) **Identifying** and splitting the words correctly. (2) **Counting** the words precisely. (3) **Planning** the word counts of each part to meet the length constraints. (4) **Aligning** the generated text with length constraints while ensuring semantic integrity.

On this basis, we conduct a decoupled error

[*]Equal contribution.
[†]Corresponding author.
[1]Our code have been released on https://github.com/chuyi369/MarkerGen.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 17370–17390
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

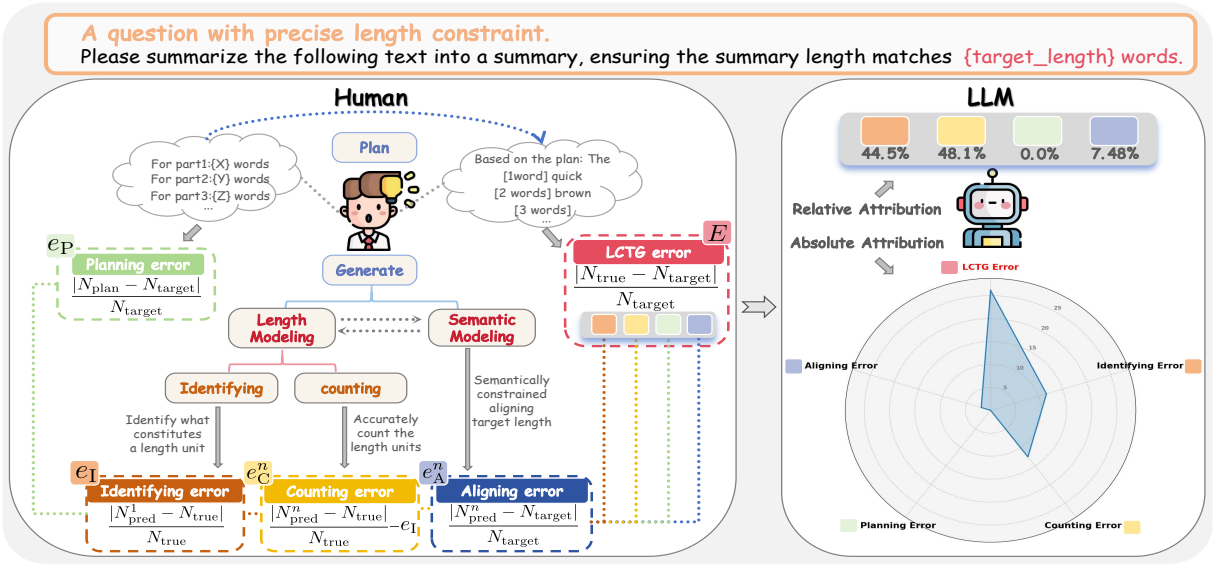


Figure 1: Sub-ability decomposition of LCTG and corresponding error analysis in LLMs.

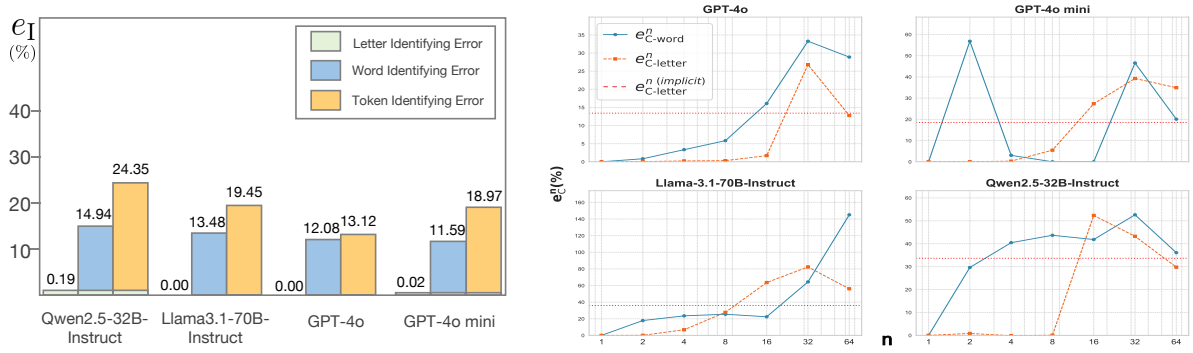
analysis of LCTG. The experimental results indicate that counting error $>$ identifying error $>$ aligning error $\gg$ planning error. This suggests that deficiencies in fundamental capabilities are the primary cause of LCTGs inferior performance. Meanwhile, it further explains why training-based approaches struggle to enhance LCTG effectively, as they are unable to provide fine-grained supervision signals for these fundamental capabilities.

Building upon this, we propose MARKERGEN, a simple-yet-effective, plug-and-play method for achieving high-quality LCTG. Specifically, to address LLMs' weaknesses in identifying and counting, we integrate external tokenizer and counter to track exact length information. To effectively convey these information to LLMs, we design an decaying interval insertion strategy that dynamically injects length markers during the generation process, enabling explicit length modeling while minimizing disruptions to semantic modeling. Furthermore, to mitigate alignment issues, we propose a three-stage decoupled generation paradigm that decouples semantic constraints from length constraints, ensuring that length constraints are better met without compromising content quality.

We conduct experiments with five LLMs on five benchmarks to validate the generalizability of MARKERGEN, covering cross-task (summarization, story generation, QA, heuristic generation), cross-scale (from 10+ to 1000+ words), cross-lingual (English and Chinese) and cross-granularity (precise and rough constraints) settings. Experimental results demonstrate that under precise length constraints, MARKERGEN reduces length errors by 12.57% compared to baselines (with an average absolute error of 5.57%), while achieving higher quality scores and incurring only 67.6% of the cost. In range-based length constraints, MARKERGEN achieves a 99% acceptance rate, further validating its effectiveness. Finally, we probe into the working mechanism of MARKERGEN through attention analysis: shallow layers primarily handle length modeling through markers, whereas deeper layers concentrate more on semantic modeling.

2 Preliminaries

We model the LCTG process of LLMs by drawing an analogy to human patterns in this task. Specifically, the model first performs content and length planning based on task requirements and length constraints. Under this plan, the semantic space expands progressively at the word level during generation, accompanied by an implicit counting process. Meanwhile, length estimation acts as a real-time constraint, dynamically regulating further extension. Ultimately, the model strives to align the length constraints while preserving semantic integrity. From this perspective, the overall **LCTG** ability of LLMs can be systematically decomposed in a bottom-up manner into **Identifying**, **Counting**, **Planning**, and **Aligning** sub-capabilities (Figure 1). Below we explore LLMs' mastery of these abilities through detailed error analysis on TruthfulQA dataset (Lin et al., 2021).



(a) Identifying error analyses

(b) Counting error analyses

Figure 2: Error analyses of fundamental abilities in LCTG across LLMs.

2.1 Identifying Error

Identifying error refers to the misidentification of fundamental length units (e.g., words), leading to discrepancies between the models estimated and actual text length. To systematically analyze this error, we instruct the model to recognize the length units of given text one by one. If we define a word as the length unit, the model should output like: "`The [1 word] quick [2 words] fox [3 words] ...`". On this basis, we calculate the identifying error rate e_{I} as follows:

$$e_{\mathrm{I}} = \frac{|N^1_{\mathrm{pred}} - N_{\mathrm{true}}|}{N_{\mathrm{true}}} \tag{1}$$

where N^1_{pred} is the models predicted final count with 1 as count interval, and N_{true} is the actual count. We subtract the error rate obtained when replacing each word with the letter "`A`" (which barely assess the identifying ability) from e_{I} to further eliminate the influence of other potential factors. We explore the word and token as length unit respectively, as shown in Figure 2a.

Finding 1. *LLMs exhibit notable e_I with both word and token as unit, showcasing their deficiencies in fundamental identifying ability.*

Finding 2. *Word yields lower e_I than token, indicating that LLMs conduct length modeling primarily based on **semantic perception** rather than **decoding mechanics**.*

2.2 Counting Error

Counting error refers to the inaccurate enumeration of length units in a given sequence, leading to deviations from the intended length. We analyze this error by prompting LLMs to count sequences with varied interval n. The case of $n = 1$ corresponds to identifying error (see §2.1). A larger n poses a greater challenge for counting accuracy.

Models	e_{P}	s_{P}	ΔE (↓)	ΔS (↑)
GPT-4o	0.06	4.28	-5.31	0.05
GPT-4o mini	0.33	3.90	+2.11	0.03
Llama-3.1-70B-Instruct	0.00	3.90	-0.63	0.04
Qwen2.5-32B-Instruct	0.04	4.22	-8.93	0.02

Table 1: e_{P} and s_{P} denote planning error and planning quality score of LLMs. ΔE and ΔS quantify the LCTG error reduction and text quality gain from two-stage generation over one-stage generation.

To decompose counting error from identifying error, we calculate e^n_{C} as follows:

$$e^n_{\mathrm{IC}} = \frac{|N^n_{\mathrm{pred}} - N_{\mathrm{true}}|}{N_{\mathrm{true}}} \tag{2}$$

$$e^n_{\mathrm{C}} = e^n_{\mathrm{IC}} - e_{\mathrm{I}} \tag{3}$$

Since LLMs exhibit negligible identifying error at the letter level (Figure 2a), error of counting letter serves as a direct measure of pure counting ability. We also include a commonly used baseline where the LLMs conduct implicit counting (directly output the length of the entire given text). The results are shown in Figure 2b.

Finding 3. ***Naive implicit counting can lead to significant errors.***

Finding 4. ***Explicit counting combined with fine-grained intervals leads to better length modeling.*** *At smaller n, the error of explicit counting is significantly lower than that of implicit counting.*

2.3 Planning Error

Planning error refers to the misallocation of word counts across different sections, leading to a discrepancy from target length. For given query and precise length constraint N_{target}, we prompt LLMs to explicitly plan both content and length

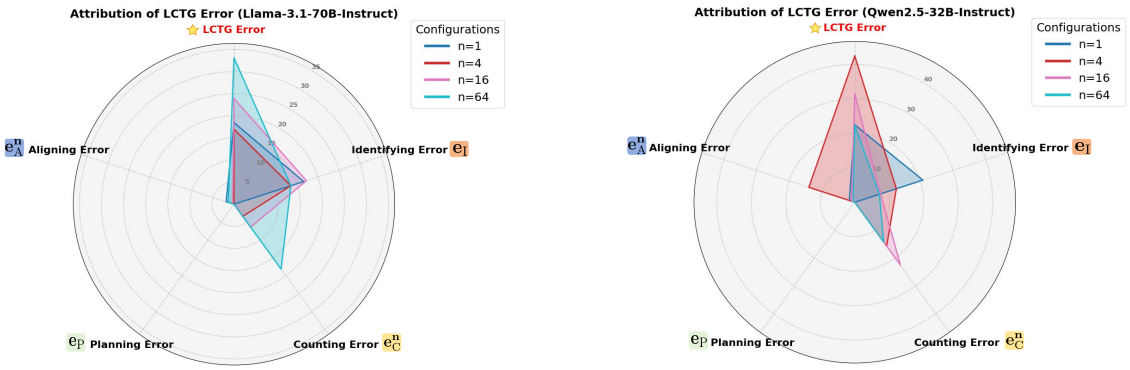


Figure 3: Absolute contribution of LCTG sub-capability deficiency on overall LCTG error across LLMs.

for each part of the response. We assess the quality [2] of the plan s_{P}, and calculate the planning error rate e_{P} as:

$$e_{\mathrm{P}} = \frac{|N_{\mathrm{plan}} - N_{\mathrm{target}}|}{N_{\mathrm{target}}} \tag{4}$$

where N_{plan} denotes the total word count allocated by the model. Meanwhile, we calculate the reduction in final length error (ΔE) and the improvement in content quality (ΔS) achieved by **planning followed by generation** compared to **direct generation**. The results are shown in Table 1.

Finding 5. ***LLMs exhibit strong planning ability.*** *The generated plan effectively meets the length constraints while achieving a quality score of around 4, demonstrating well-structured content allocation.*

Finding 6. ***Planning before generation brings better results.*** *Compared to direct generation, executing planning and generating sequentially for decomposition reduces length deviations while enhancing semantic quality.*

2.4 Aligning Error

Aligning error refers to the discrepancy between the models perceived length and the target length, arising from the challenge of maintaining semantic integrity while adhering to length constraints. We calculate aligning error as follows:

$$e_{\mathrm{A}}^{n} = \frac{|N_{\mathrm{pred}}^{n} - N_{\mathrm{target}}|}{N_{\mathrm{target}}} \tag{5}$$

where N_{pred}^{n} represents the models perceived length with counting interval n, i.e., the length the model assumes it has generated. We calculate and show the e_{A}^{n} in Figure 4.

[2] We use Qwen-Plus (Yang et al., 2024) as the judge with a scoring range of [1, 5]. See corresponding prompts in Appendix B.3

Finding 7. ***Smaller counting intervals introduce greater aligning error.*** *By closely analyzing cases, we find that frequent explicit counting interferes with semantic modeling, causing early termination of generation and poor alignment. In contrast, larger length intervals approximate implicit counting, preserving a more natural generation process.*

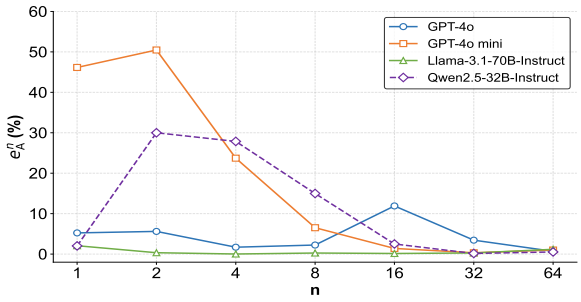


Figure 4: Aligning Error across varied length intervals.

2.5 LCTG Error

LCTG error refers to the discrepancy between the actual length of generated text and the target length:

$$E = \frac{|N_{\mathrm{true}} - N_{\mathrm{target}}|}{N_{\mathrm{target}}} \tag{6}$$

As established above, this error is systematically composed of four components: **Identifying Error** (§2.1), **Counting Error** (§2.2), **Planning Error** (§2.3), and **Aligning Error** (§2.4). To investigate the key factors influencing LLMs' LCTG error, we calculate their absolute contributions $\dot{e}_i^n$ under different length interval n as follows:

$$\dot{e}_i^n = \frac{e_i^n}{e_{\mathrm{I}} + e_{\mathrm{C}}^n + e_{\mathrm{P}} + e_{\mathrm{A}}^n} \times E^n, \quad i \in [\mathrm{I,C,P,A}] \tag{7}$$

The results are shown in Figure 3. Further details can be found in B.

Finding 8. ***LCTG error is primarily attributed to fundamental deficiencies in length modeling, fol-***

lowing the order of Counting Error > Identifying Error > Aligning Error ≫ Planning Error. *Thus, as counting interval increases, the accumulation of counting errors leads to a corresponding rise in LCTG error.*

3 Methodology

Based on the analyses and findings above, we propose MARKERGEN, a simple-yet-effective plug-and-play method to help LLMs attain better LCTG performance, as shown in Figure 5. This method consists of two key modules: (1) **Auxiliary Marker Insertion Decoding** mechanism, which explicitly enhances length modeling during generation; (2) **Three-Stage Decoupled Generation** scheme, which decouples length constraints from semantic content generation to further improve LCTG performance.

3.1 Auxiliary Marker Insertion Decoding

External Tool Invocation. Our analysis in §2 reveals that LLMs exhibit significant identifying and counting errors, which directly contribute to inaccuracies in length modeling. To mitigate these fundamental deficiencies, we introduce external tokenizer and counter for unit recognition and counting, respectively. As Finding 1 indicates that LLMs perceive words better than tokens, we select words as the length unit.

Length Information Injection. With precise length information, we consider feeding it into the model for length modeling. Since Finding 3 indicates that LLMs' inherent implicit length modeling leads to significant errors and is inconvenient for incorporating external length information, we actively insert precise length markers during generation to enable explicit length modeling:

$$\text{Len}(x) = \text{Counter}(\text{Tokenizer}(x))$$

$$x_{t+1} = \begin{cases} \text{Marker}(\text{Len}(x_{\leq t})), & \text{if } \mathcal{S}(\text{Len}(x_{\leq t}), N) \\ \text{Sampling}(P(x_{t+1}|x_{\leq t})), & \text{else} \end{cases} \tag{8}$$

where $P(x_{t+1}|x_{\leq t})$ is the LLM's probability distribution for next token, Marker defines the marker format (e.g., [20 words], we discuss the effects of varied marker formats in Appendix C.1), $\mathcal{S}$ is the strategy that determines whether to insert a marker based on current length $\text{Len}(x)$ and target length N. By treating the inserted markers as anchors, LLMs can continuously adjust the expected length of content to be generated during the generation process, thereby reducing the final LCTG error.

Decaying Interval Marker Insertion Strategy. The most naive insertion strategy involves placing markers at uniform intervals, which we denote as $\mathcal{S}_{uni}$. However, according to Findings 4 and 7, a smaller insertion interval n improves length modeling but compromises semantic modeling, whereas a larger n exhibits the opposite effect. Considering this, we propose a strategy $\mathcal{S}_{dec}$, where n decays exponentially during the generation process:

$$\mathcal{S}_{dec}(x, N) = \begin{cases} \text{True}, & \text{if } x \in \{N - \text{int}(2^{-i} \times N)\}_{i \in \mathbb{N}} \\ \text{False}, & \text{else} \end{cases} \tag{9}$$

Taking $N = 200$ as an example, the maker will be inserted behind the 100th, 150th, 175th, ... words. At the early stage of generation, the model primarily focuses on semantic modeling. As the generation progresses, it increasingly emphasizes length control, ultimately leading to a smaller LCTG error. Consequently, $\mathcal{S}_{dec}$ effectively balances semantic modeling and length modeling.

3.2 Three-Stage Decoupled Generation

Finding 7 validates that aligning error primarily arises from the inferior semantic modeling, which causes premature termination of the generation process. While the planning before generation scheme alleviates interference in semantic modeling by decoupling the planning process (Finding 6), it still entangles length modeling with semantic modeling. To mitigate this, we introduce a three-stage decoupled generation scheme to further reduce the alignment error and improve the text quality, as illustrated in Figure 5.

Stage One: Planning. The model generates a reasonable plan based on the input query and length constraints, including the content of each section and the word allocation.

Stage Two: Ensuring Semantic Integrity. The model focuses on semantic modeling to generate a high-quality response per the plan without being strictly required to adhere to length constraints.

Stage Three: Aligning Length Constraints. Responses generated in stage two are usually of high quality but may not meet length restrictions. To refine them, we use these non-compliant responses as input and apply the Auxiliary Marker Insertion Decoding mechanism for rewriting. The **rewriting requirements** include: (1) Retaining the high-quality semantic modeling of the input

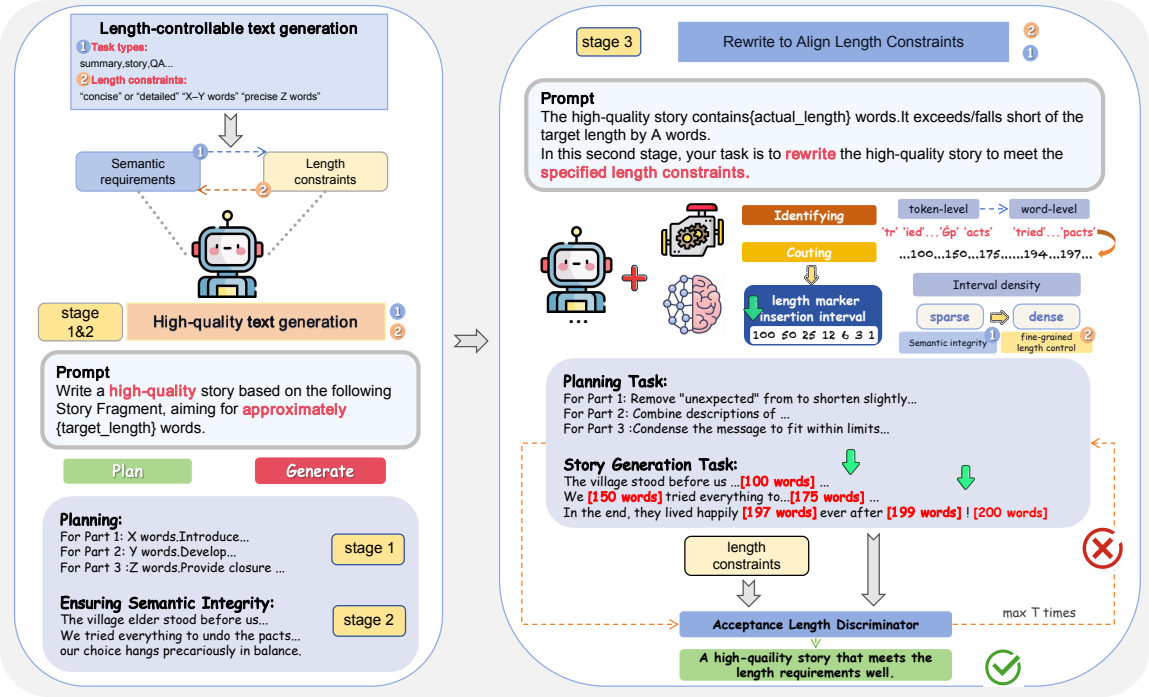


Figure 5: Overview of MARKERGEN.

Benchmarks	Ability Tested	Length (words)
CNN/DailyMail (Nallapati et al., 2016)	Summarization	18-165
HANNA (Chhun et al., 2022)	Story Generation	139-995
TruthfulQA (Lin et al., 2021)	Question Answering	101-294
HelloBench (Que et al., 2024)	Heuristic LCTG& Open-ended QA	489-1450
GAOKAO (Zhang et al., 2023b)	History Open-ended QA	71-901

Table 2: Benchmarks Introduction.

content. (2) Strictly adhering to the specified length constraints. In terms of **workflow**, the model is required to: (1) Firstly analyze the previous stage's response for potential improvements; (2) If its output does not meet the length constraints, it will be regenerated up to T times or until the constraints are met.

See Appendix F for prompts of each stage.

4 Experiments

We conduct comprehensive experiments to examine MARKERGEN. Specifically, we validate its effectiveness in §4.2, analyze its generalizability in §4.3, explore the impact of its key components in §4.4, and provide further insights into its mechanism in §4.5. Hyperparameter choices and additional analyses are provided in Appendix E.

4.1 Experimental Settings

Benchmarks We choose five benchmarks for experiments, where HelloBench includes two subsets, as shown in Table 2. See details in Appendix D.

Baselines

- **Ruler** (Li et al., 2024b): A training-based[3] method that defines length control templates to regulate generation at the range level.
- **Implicit** (Bai et al., 2024): Conduct a plan-and-generate process without explicit counting. To ensure a fair comparison, the model generates multiple responses until token count outperforms MARKERGEN and the candidate with the smallest LCTG error is selected.

Details We conduct extensive experiments using Qwen2.5 series (Qwen2.5-7B/14B/32B-Instruct) (Yang et al., 2024) and the Llama3.1 series (Llama-3.1-8B/70B-Instruct) (Dubey et al., 2024), with sampling temperature as 0.5. We experiment under coarse-grained length constraints on the Open-ended QA subset of HelloBench and assess the LCTG error rate under precise length constraints on other benchmarks, following Eq. (6). To evaluate the text quality, we use GPT-4o mini (Hurst

[3]Ruler is the only training-based baseline for which we can find that releases the code and training set.

Benchmarks	Methods	Qwen2.5 Series						Llama3.1 Series				Costs
		7B		14B		32B		8B		70B		
		E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	
CNN/DailyMail	Implicit	30.31	3.04	12.54	3.15	11.05	3.21	15.12	3.04	11.07	3.09	$1.30 \times \delta$
	MARKERGEN	**9.92**	**3.07**	**6.06**	**3.16**	**4.82**	**3.25**	**3.36**	**3.18**	**3.18**	**3.36**	δ
HANNA	Implicit	28.55	3.47	14.86	3.55	12.03	3.67	16.68	3.54	10.44	3.61	$2.37 \times \delta$
	MARKERGEN	**8.49**	**3.50**	**5.22**	3.55	**3.57**	**3.72**	**2.98**	**3.60**	**2.58**	**3.63**	δ
TruthfulQA	Implicit	16.7	4.29	17.9	4.44	8.7	4.45	7.21	4.22	7.64	4.46	$1.75 \times \delta$
	MARKERGEN	**9.08**	**4.33**	**7.59**	4.43	**4.48**	**4.54**	**3.82**	**4.25**	**2.80**	**4.48**	δ
Heuristic Generation	Implicit	35.69	3.42	21.34	3.80	12.02	3.80	21.91	3.72	27.89	3.74	$1.06 \times \delta$
	MARKERGEN	**8.51**	**4.13**	**6.35**	**4.00**	**5.34**	**4.14**	**6.03**	**4.03**	**5.03**	**3.98**	δ

Table 3: Overall Performance of MARKERGEN on Various Benchmarks. E denotes LCTG error rate (%) and S denotes the text quality ([1, 5]) given by LLM judge. δ denotes the token cost of MARKERGEN under each setting.

Model	Methods	Target Length Scales								Costs
		100		200		300		400		
		E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	
Qwen2.5-7B-Instruct	Implicit	30.97	3.45	22.91	3.53	26.12	3.28	29.63	3.08	$1.26 \times \delta$
	MARKERGEN	8.26	3.92	9.06	4.00	7.67	3.75	5.10	3.55	δ
Model	**Methods**	**Length Constraint Types**								**Costs**
		<100		100-150		160-200		>500		
		E_r (↓)	S (↑)	E_r (↓)	S (↑)	E_r (↓)	S (↑)	E_r (↓)	S (↑)	
Qwen2.5-7B-Instruct	Implicit	7.50	3.47	63.00	4.03	66.00	4.06	29.50	2.65	$1.07 \times \delta$
	MARKERGEN	0.00	3.94	0.50	4.50	3.00	4.53	0.00	3.13	δ

Table 4: Experiments with varied length scales and constraint types on Open-ended QA subset of HelloBench.

et al., 2024) as the judge, with a calibration algorithm to mitigate the length bias (Zheng et al., 2023) (See details in Appendix E). For precise constraints, we set the length of ground truth response as desired target length. We run each setting for three times and report the average results.

4.2 Main Results

As shown in Table 3, the commonly used two-stage implicit counting baseline results in a substantial LCTG error rate E of 18.32% on average, even if the best response is chosen across multiple attempts. This intuitively demonstrates the impact of the inherent limitations of LLM's LCTG sub-capability. The training-based baseline Ruler, as observed in our preliminary experiments (Appendix E.1), benefits from training on test sets that matches the training domain, while performs poorly on our evaluated benchmarks, highlighting its limited generalizability. In comparison, under strict length constraints, MARKERGEN achieves an absolute reduction of 12.57% in E relative to the implicit baseline, bringing the final error down to just 5.57%. In terms of text quality, by decoupling length modeling and semantic modeling during the generation process and employing the decaying insertion strategy to minimize the damage caused by length constraints to semantic integrity, MARKERGEN achieves a higher S in average. Meanwhile, this performance is achieved with only 64% of the tokens used by the baseline.

4.3 Generalizability

Across LLMs and Tasks. Table 3 demonstrates the strong generalizability of MARKERGEN to LLMs and generation tasks.

Across Length Scale. Table 3 also shows MARKERGENs strong performance across benchmarks with varying length scale (18-1450). To further investigate, we analyze progressively increasing the target length from 100 to 400. The results in Table 4 show a declining trend in MARKERGENs error rate, which can be attributed to the auxiliary marker insertion decoding mechanism that prevents error accumulation from implicit modeling.

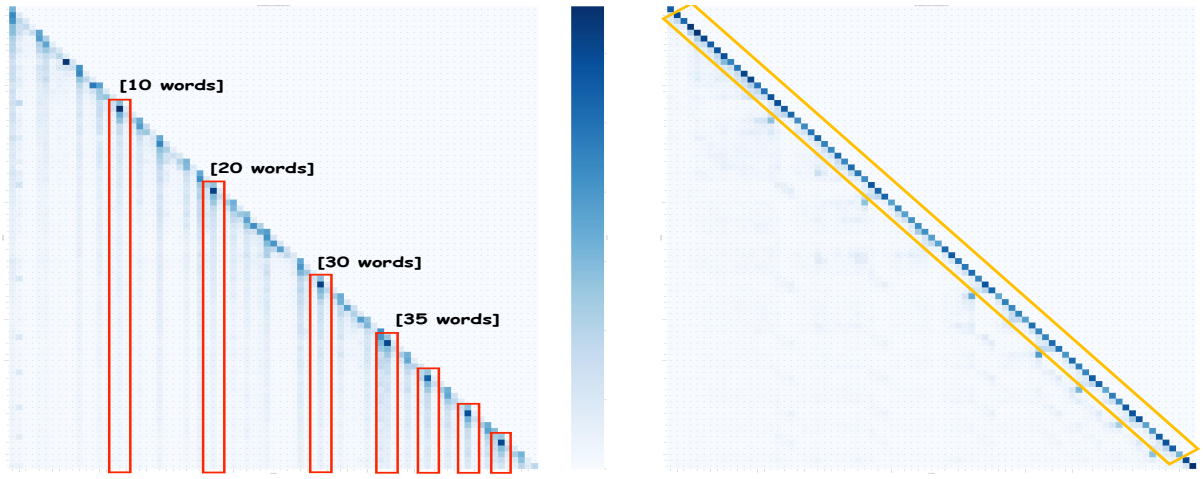


Figure 6: Attention matrices of the first (left) and last (right) layers.

Variants	Marker Insertion Interval n											
	1		4		16		32		64		Decaying	
	E (↓)	S (↑)	E (↓)	S (↑)	E	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)	E (↓)	S (↑)
w/o Tool	15.53	4.28	32.50	4.29	34.64	4.46	32.50	4.48	20.44	4.58	–	–
Two Stage	3.10	4.03	1.49	4.03	4.04	4.20	3.26	4.23	3.93	4.32	2.66	4.28
Three Stage	4.84	4.28	4.20	4.29	4.89	4.45	5.45	4.48	5.18	4.57	4.48	4.54

Table 5: Ablation studies on key components.

Across Constraint Types. In addition to exact length constraints, users may impose range-based limits. We evaluate E_r[4], the proportion of responses violating these constraints. Table 4 shows that MARKERGEN maintains an E_r below 3% in all cases, significantly lower than the baseline.

Across Lingual. We further validate the effectiveness of MARKERGEN in Chinese setting on GAOKAO benchmark, as shown in Table 8.

4.4 Ablation Studies

In this section, we validate the effectiveness of each module in MARKERGEN with Qwen2.5-32B-Instruct on TruthfulQA, as shown in Table 5.

Tool Invocation. When the model is required to insert markers independently without relying on an external tokenizer and counter, its fundamental limitations lead to a significant increase in the error rate, exceeding 15%.

Decaying Interval Marker Insertion. When using a fixed marker insertion interval n, since length control is inversely proportional to n, while semantic modeling is directly proportional to n (which induces alignment errors), we observe unstable LCTG error rate. In contrast, by adopting a sparse-to-dense insertion approach, the Decaying Interval Marker Insertion strategy ensures explicit length modeling while maximizing semantic integrity, leading to lower E and superior S.

Three-Stage Decoupled Generation. The introduction of explicit length markers in the two-stage scheme leads to a substantial reduction in LCTG error relative to the implicit baseline ($8.7 \rightarrow 2.66$). However, this scheme places greater emphasis on length modeling, which consequently diminishes text quality ($4.45 \rightarrow 4.28$). In comparison, the three-stage scheme achieves a better balance by decoupling semantic and length modeling, thereby improving both length control and text quality.

4.5 Working Mechanism of MARKERGEN

To better understand how LLMs leverage the inserted length markers in MARKERGEN, we visualize the attention matrices of the first and last layers of Llama-3.1-8b-Instruct (Figure 6). In the shallow layers, the attention distribution reveals a clear focus on the length information represented by the length markers (in the red box). As the model progresses to the deeper layers, attention shifts from the length information to the adjacent semantic content (in the orange box). This pattern demonstrates that at shallow layers, the model uses markers to establish length modeling and encode pre-

[4] $E_r = 1 - \frac{|\{N_{\text{true}} | N_{\text{target}}^{\min} \leq N_{\text{true}} \leq N_{\text{target}}^{\max}\}|}{N_{\text{total}}}$.

cise length information. At deeper layers, it relies on this length information for semantic modeling, producing tokens that align with the length constraints while maintaining semantic integrity.

Conclusions

To improve the performance of LLMs in length-controllable text generation, we conduct a bottom-up error analysis of relevant sub-abilities. The results reveal that deficiencies in identifying, counting, and aligning are key limitations. To fill this gap, we propose MARKERGEN, which leverages external tools to compensate for fundamental deficiencies. Additionally, it introduces Decaying Interval Marker Insertion Strategy to facilitate explicit length modeling and employs Three-Stage Decoupled Generation mechanism to balance semantic coherence and length control. Comprehensive experiments demonstrate the strong generalizability and effectiveness of MARKERGEN in enhancing length control and preserving semantic integrity.

Limitations

In this work, we conduct a bottom-up sub-capability analysis in the LCTG ability and propose the MARKERGEN method, achieving strong LCTG performance. One major limitation of MARKERGEN is that it is currently only applicable to open-source models and cannot yet be used with closed-source models. To address this, we will release our code, allowing closed-source model providers interested in adapting MARKERGEN to benefit from our method in enhancing LCTG performance.

Ethics Statement

All of the datasets used in this study were publicly available, and no annotators were employed for our data collection. We confirm that the datasets we used did not contain any harmful content and was consistent with their intended use (research). We have cited the datasets and relevant works used in this study.

Acknowledgments

This work is supported by Beijing Natural Science Foundation (No.4222037, L181010).

References

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longwriter: Unleashing 10,000+ word generation from long context llms. *arXiv preprint arXiv:2408.07055*.

Bradley Butcher, Michael O'Keefe, and James Titchener. 2024. Precise length control in large language models. *arXiv preprint arXiv:2412.11937*.

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

Cyril Chhun, Pierre Colombo, Chloé Clavel, and Fabian M Suchanek. 2022. Of human criteria and automatic metrics: A benchmark of the evaluation of story generation. *arXiv preprint arXiv:2208.11646*.

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

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

Kiannah Foster, Andrew Johansson, Elizabeth Williams, Daniel Petrovic, and Nicholas Kovalenko. 2024. A token-agnostic approach to controlling generated text length in large language models.

Google. 2024. Gemini 2.0 Flash. https://deepmind.google/technologies/gemini/flash/.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024a. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Yuxuan Gu, Wenjie Wang, Xiaocheng Feng, Weihong Zhong, Kun Zhu, Lei Huang, Tat-Seng Chua, and Bing Qin. 2024b. Length controlled generation for black-box llms. *arXiv preprint arXiv:2412.14656*.

Chenyang Huang, Hao Zhou, Cameron Jen, Kangjie Zheng, Osmar R ZaÃ■ane, and Lili Mou. 2025. A decoding algorithm for length-control summarization based on directed acyclic transformers. *arXiv preprint arXiv:2502.04535*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Renlong Jie, Xiaojun Meng, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Prompt-based length controlled generation with reinforcement learning. *CoRR*, abs/2308.12030.

Juseon-Do Juseon-Do, Hidetaka Kamigaito, Manabu Okumura, and Jingun Kwon. 2024. Instructcmp: Length control in sentence compression through instruction-based large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 8980–8996.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2024. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al. 2024a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Jiaming Li, Lei Zhang, Yunshui Li, Ziqiang Liu, Yuelin Bai, Run Luo, Longze Chen, and Min Yang. 2024b. Ruler: A model-agnostic method to control generated length for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 3042–3059. Association for Computational Linguistics.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024. Controllable text generation for large language models: A survey. *CoRR*, abs/2408.12599.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *CoRR*, abs/2109.07958.

Sangjun Moon, Jingun Kwon, Hidetaka Kamigaito, Manabu Okumura, et al. Length representations in large language models.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 4998–5017. Association for Computational Linguistics.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, et al. 2024. Hellobench: Evaluating long text generation capabilities of large language models. *arXiv preprint arXiv:2409.16191*.

Fabian Retkowski and Alexander Waibel. 2024. Zero-shot strategies for length-controllable summarization. *arXiv preprint arXiv:2501.00233*.

Seoha Song, Junhyun Lee, and Hyeonmok Ko. 2024. Hansel: Output length controlling framework for large language models. *arXiv preprint arXiv:2412.14033*.

Zekun Wang, Feiyu Duan, Yibo Zhang, Wangchunshu Zhou, Ke Xu, Wenhao Huang, and Jie Fu. 2024. Positionid: Llms can control lengths, copy and paste with explicit positional awareness. *arXiv preprint arXiv:2410.07035*.

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

Peiwen Yuan, Shaoxiong Feng, Yiwei Li, Xinglin Wang, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025. Llm-powered benchmark factory: Reliable, generic, and efficient. *arXiv preprint arXiv:2502.01683*.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2024. Following length constraints in instructions. *CoRR*, abs/2406.17744.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023a. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023b. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, 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: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Related Work

LCTG Methods Text length is a fundamental aspect of natural language that carries semantic information, making LCTG a task of balancing length and semantic constraints. Achieving precise length control remains a challenge for LLMs due to limitations in their architecture, such as position encoding (Butcher et al., 2024; Kazemnejad et al., 2024; Chang and Bisk, 2024) and decoding mechanisms(Huang et al., 2025). Consequently, existing methods focus on injecting length information to help LLMs model length accurately, which can be categorized into training-based and inference-based approaches.

Training-based methods inject varying levels of length signals during fine-tuning or reinforcement learning. For instance, Jie et al. (2023); Li et al. (2024b) use prompt templates to teach LLMs the mapping between length and textual content, while Song et al. (2024); Wang et al. (2024) design fine-grained datasets to guide correct length modeling. Other methods, like Yuan et al. (2024); Jie et al. (2023), utilize reward functions to align length preferences during training. While effective in certain scenarios, these methods suffer from limited generalization across diverse LCTG tasks, including varying length constraints and instructions.Inference-based methods adjust inputs multiple times during generation to inject, such as through prompt-based Automated Revisions and Sample Filtering (Retkowski and Waibel, 2024; Juseon-Do et al., 2024), or length-controlled importance sampling during decoding (Gu et al., 2024b). Although these approaches can better generalize length alignment, they still struggle with achieving precise control.

While both approaches enhance LCTG, they often apply a top-down strategy that lacks deep understanding and targeted enhancement of LCTG sub-capabilities. This limits progress in meeting length constraints accurately. Furthermore, many methods neglect semantic constraints, and injecting length information may degrade text quality. Therefore, we propose MARKERGEN to bridge this gap for precise length control and preserving semantic integrity.

B Detailed Sub-ability Error analyses in LCTG

B.1 Identifying Error

Identifying error refers to the misidentification of fundamental length units. To systematically analyze this error, we design a counting experiment in which the model is prompted to sequentially recognize and accumulate length units, then compare its predicted count with the ground truth. Experimental results confirm that in the one-by-one accumulation setting, counting errors do not occur, meaning that the final length error entirely arises from identifying error (as shown in Figure 7).

B.2 Counting Error

Counting error refers to the inaccurate enumeration of units in a given sequence, leading to deviations from the intended length. Therefore, in the setting where $n > 1$ in the counting experiment, the final counting result error is caused by both identifying error and counting error. In this case, counting error can be decoupled by resolving identifying errors in the accumulation process, where errors result from the accumulation step.We also conducted the same counting experiment as in Section 2.2 on the CNN/DailyMail summarization dataset, as shown in Figures 8.

From the figure, we can further validate the same conclusions as in Findings 1, 2, 3, and 4 in Section 2, revealing that the length errors in the generated results of the LCTG task stem from significant errors in the LLM's perception and modeling of length.

B.3 Planning Error

Planning error refers to the misallocation of word counts across different sections, leading to a discrepancy from target length.The planning ability of LLMs encompasses not only length planning but also semantic planning. To effectively assess the quality of LLMs semantic planning, we use Qwen-Plus (Yang et al., 2024) as the judge, with a scoring range of [1, 5]. The specific evaluation prompt is as follows:

> You are tasked with evaluating the quality of a generated answer plan for a TruthfulQA question. The evaluation should focus on the truthfulness, logical coherence, and adherence to the given prompt and instructions. Rate the answer plan on a 5-point scale as follows:

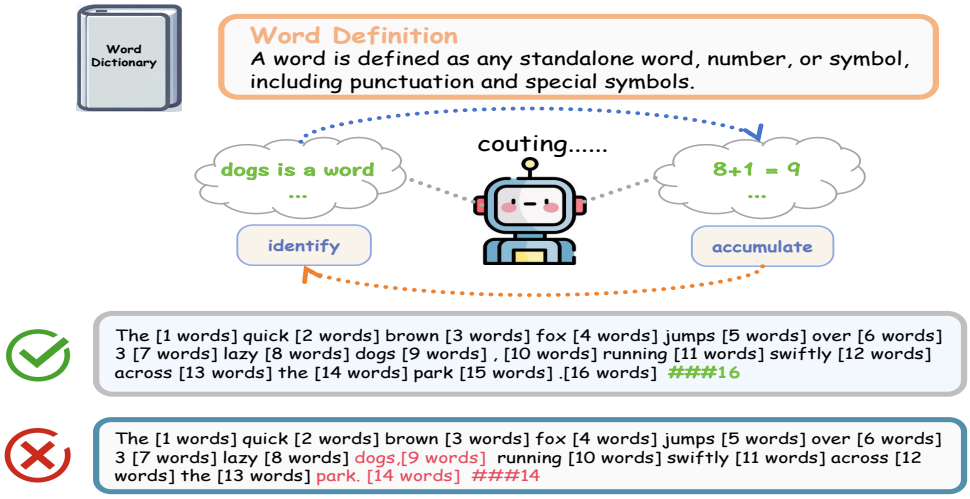


Figure 7: Schematic diagram of counting experiment under the condition of $n = 1$

- **5: Outstanding** - The plan is highly truthful, logically coherent, and perfectly adheres to the prompt and instructions.
- **4: Very Good** - The plan is mostly truthful and coherent, with only minor issues in details or adherence to instructions.
- **3: Good** - The plan is acceptable but has noticeable shortcomings in truthfulness or coherence.
- **2: Poor** - The plan has significant issues in truthfulness or logical coherence and does not adequately follow the instructions.
- **1: Unacceptable** - The plan is largely untruthful, incoherent, or fails to follow the prompt instructions entirely.

Please provide the overall score in the following format: `###score X`

Question:

`+ prompt`

Generated Answer Plan:

`+ generated_plan`

Evaluate the answer plan based on the above criteria.

Since the LCTG task requires meeting both length and semantic constraints, utilizing the LLM's superior planning ability for explicit planning before generation, as opposed to direct generation, helps to clearly define the modeling space for length and the semantic extension range. This not only contributes to improved text generation quality but also reduces length errors.

B.4 Aligning Error

Aligning error refers to the discrepancy between the models perceived length and the target length, arising from the challenge of maintaining semantic integrity while adhering to length constraints.As shown in Figure 4, aside from Finding 7, we observe significant differences in aligning error across models. Qwen2.5-32B-Instruct and GPT-4o mini exhibit larger alignment errors under fine-grained length modeling. As discussed in Section 2, length estimation acts as a real-time constraint, dynamically regulating further extension. Ultimately, the model strives to align the length constraints while preserving semantic integrity. High-frequency length perception updates pose greater challenges for the natural expansion of the semantic space, which explains why some models with weaker robustness in semantic expansion show significant alignment errors. These errors become a primary source of LCTG inaccuracies (as shown in Figure 9). This further emphasizes that LCTG is a task of balancing length and semantic constraints.

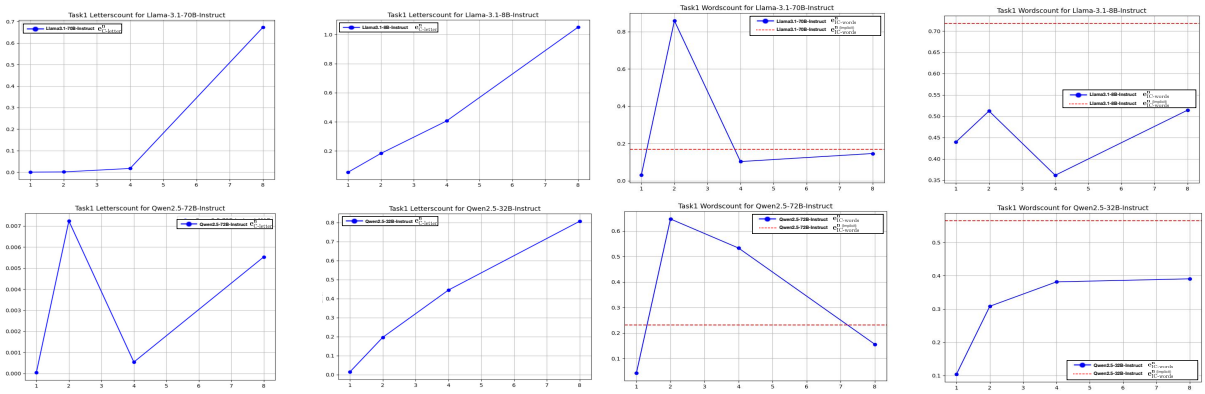


Figure 8: Error analyses of fundamental abilities in LCTG on CNN/DailyMail.

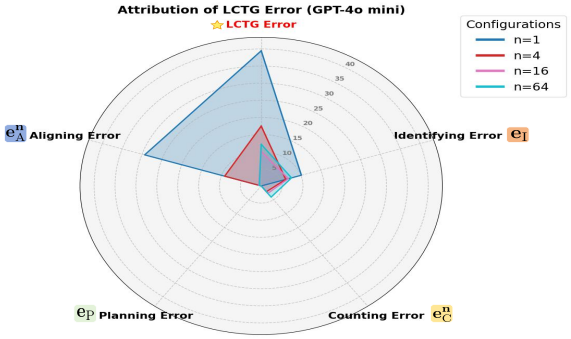


Figure 9: Absolute contribution of LCTG sub-capability deficiency of GPT 4o-mini.

B.5 LCTG Error

Based on the above decomposition of sub-abilities in LCTG and the corresponding error analysis, we can clearly quantify the contribution of each decoupled error to the final LCTG error. As shown in Figure 1, the quantification results in the right figure represent the average values of four models under various n conditions. The conclusion we can draw is that the primary cause of significant length errors in current mainstream LLMs on LCTG tasks is the lack of bottom-up identification and counting capabilities required for accurate length modeling.

C Exploration of Interval Marker Insertion Strategy Variants

C.1 Length Marker Forms

We explored the impact of different forms of length marker insertion on performance, such as the number of words generated "[k]", the semantic marker "[k words]", and the remaining words to be generated "[N_{target} - k]" (remaining words). As shown in Table 6, we used Llama-3.1-8B-Instruct on the CNN/DailyMail dataset to investigate the effects of various marker forms under multiple n conditions on generation error and text quality. The results show that using a semantic length marker representing the number of words generated achieved the best performance in both length error and text quality.

Marker Form	E (↓)	S (↑)
$[k]$	18.28	3.10
$[k$ words$]$	**15.74**	**3.14**
$[N_{\text{target}} - k]$	27.92	3.09

Table 6: Comparison of Length Marker Forms and Their Performance

D Detailed Benchmarks Introduction

The benchmarks used in our experiments are as follows:

- **CNN/DailyMail**(Nallapati et al., 2016): A summarization dataset of news articles, with 500 randomly sampled items. (*18-165 words*)
- **HANNA**(Chhun et al., 2022): A long-form story generation dataset with 200 selected items. (*139-995 words*)
- **TruthfulQA**(Lin et al., 2021): A benchmark for factual accuracy in open-domain QA. (*101-294 words*)
- **HelloBench**(Que et al., 2024): A long-text generation benchmark. We selected subsets from *heuristic text generation* (e.g., argumentative and roleplaying writing, covering five types) and *open-ended QA* (spanning ten domains). (*489-1450 words*)

- **GAOKAO-Bench**(Zhang et al., 2023b): A benchmark collected from the Chinese college entrance examination (GAOKAO). We selected the *2010-2022 History Open-ended Questions* subset. (*71-901 words*)

E Detailed Experimental Results

E.1 Performance and Generalization Study of Training-based Methods

To investigate the performance and generalization of training-based methods in diverse, real-world LCTG task scenarios, we selected Ruler, a training-based method that defines length control templates to regulate generation at the range level. This choice is based on the fact that Ruler is the only training-based baseline for which the code and training set are publicly available. We followed the exact setup provided in the repository and verified the correctness of our replication by achieving significant performance improvements on the given test set, as shown in Table 7.

The test set of Ruler adopts two custom evaluation metrics: Precise Match (PM) and Flexible Match (FM). PM requires the output length to fall within a narrow target interval, while FM allows a broader tolerance range. The metrics are defined as:

$$PM = \frac{1}{N}\sum_{i=1}^{N} 1\left(\text{lb}^{P}_{\text{TL}_i} < L(c_i) \leq \text{ub}^{P}_{\text{TL}_i}\right)$$

$$FM = \frac{1}{N}\sum_{i=1}^{N} 1\left(\text{lb}^{F}_{\text{TL}_i} < L(c_i) \leq \text{ub}^{F}_{\text{TL}_i}\right)$$

where $L(c_i)$ is the length of the i-th generated output, and TL_i denotes the corresponding target length.

Next, we tested the trained model, referred to as Llama-3.1-8B-Instruct-ruler, across four selected benchmarks with varying tasks, length scales, and instructions, under cost-alignment conditions. The experimental results revealed substantial errors and a decline in text quality, even when compared to the implicit method's results without training (as shown in Table 3). This finding demonstrates the limited generalization capability of the method, highlighting its struggle to cope with the complexity and diversity of real-world LCTG scenarios.

E.2 Length Bias Correction in LLMs-as-a-Judge

It has been demonstrated that LLMs-as-a-judge exhibit a noticeable length bias (Li et al., 2024a; Gu et al., 2024a). To evaluate the quality of generated text objectively and accurately for LCTG tasks, it is essential to correct for this length bias. We adopt the length-controlled AlpacaEval (Dubois et al., 2024) and Yuan et al. (2025).

To derive unbiased judge scores, we use a Multiple Regression model. Specifically, we set the judge score as the dependent variable, with the generator categories as dummy variables, and the length of the generated text as a covariate. The model is formulated as follows:

$$f(i) = \beta_0 + \beta_M \cdot C(\text{Method}) + \beta_\text{m} \cdot C(\text{Model}) + \beta_\text{l} \cdot \text{Length} + \epsilon \tag{10}$$

Where $f(i)$ denotes the judge score for the generated text $\mathcal{G}_i$, $C(\text{Method})$ and $C(\text{model})$ are categorical variables representing the method and the model used, respectively, and Length is the actual length of the generated text. The coefficients β_M, β_m, and β_l are used to adjust the raw judge score $f(i)$, effectively removing length bias by setting it to zero. These adjusted scores, free from length bias, serve as the metrics for faithfulness and alignment.

E.3 Win-rate and LLM-as-Judge Based Evaluation

To provide a more robust and trustworthy evaluation of generation quality, we complement our main evaluation protocol with a win-rate-based assessment and multi-LLM judgment analysis. Specifically, we adopt the pairwise evaluation framework from AlpacaEval 2.0 (Dubois et al., 2024), using GPT-4o-mini as the automatic judge. The generated outputs from the Implicit baseline and our proposed MARKERGEN method are compared against reference model Gemini-2.0 Flash(Google, 2024) across four benchmarks. As shown in Table 9, MARKERGEN consistently achieves higher win rates across most configurations, verifying the effectiveness of the Three-Stage Decoupled Generation in improving response quality.

To further confirm the reliability of using LLMs as judges, we conducted a human agreement study. We randomly sampled 200 pairwise comparisons and observed a strong Spearman's correlation co-

TLG dataset				
benchmark	Method	Models	PM (↑)	FM (↑)
TLG dataset	before training	Llama-3.1-8B-Instruct	5.55	10.20
	RULER	Llama-3.1-8B-Instruct-ruler	41.75	55.10
Precise Length Constraint Benchmarks				
Benchmarks	Methods	Models	E (↓)	S (↑)
CNN/DailyMail	Ruler	Llama-3.1-8B-Instruct-ruler	78.21	3.10
	MARKERGEN	Llama-3.1-8B-Instruct	3.36	3.18
HANNA	Ruler	Llama-3.1-8B-Instruct-ruler	68.21	2.87
	MARKERGEN	Llama-3.1-8B-Instruct	2.98	3.60
TruthfulQA	Ruler	Llama-3.1-8B-Instruct-ruler	44.93	3.27
	MARKERGEN	Llama-3.1-8B-Instruct	3.82	4.25
Heuristic Generation	Ruler	Llama-3.1-8B-Instruct-ruler	66.17	2.94
	MARKERGEN	Llama-3.1-8B-Instruct	6.03	4.03

Table 7: Evaluation of the training-based method RULER on its own test set and four generalization benchmarks.

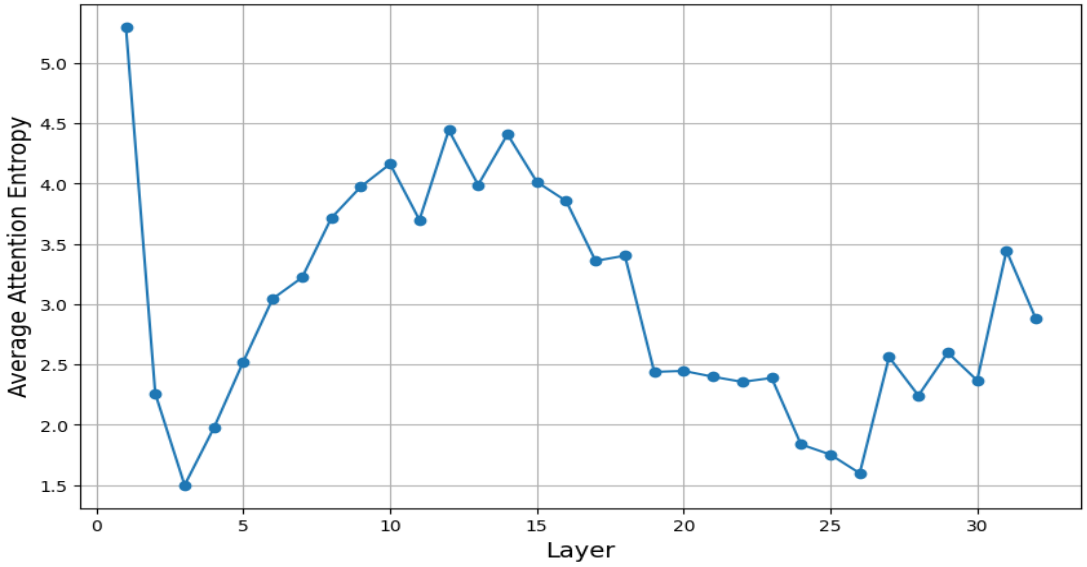


Figure 10: Attention Entropy across layers.

Methods	**Models**	E (↓)	S (↑)
Implicit	Qwen2.5-14B-Instruct	27.41	3.47
MARKERGEN	Qwen2.5-14B-Instruct	7.71	3.55

Table 8: GAOKAO-History Chinese Dataset Results

efficient of 0.8794 between GPT-4o-mini judgments and human annotations, indicating high consistency.

In addition, we report point-wise quality scores evaluated by three different LLMs: GPT-4o, Gemini-2.0-Flash (`thinking-exp`), and GPT-4o-mini. Tables 10, 11, 12, and 13 summarize these results across the CNN/DailyMail, HANNA, TruthfulQA, and Heuristic Generation benchmarks, respectively. Each table reports the average score (S_{avg}), variance (σ^2), and model sizes. MARKERGEN consistently improves average scores across all model sizes and datasets, while maintaining comparable or lower variance, further reinforcing its robustness and generalizability.

E.4 Enhanced Ablation Study

To provide a more comprehensive validation of our method, we extend the original component ablation study (Table 5) by evaluating additional combinations of stages. This enhanced analysis is motivated by the need to isolate and quantify the individual and joint effects of the key stages, as suggested in the review. Table 14 presents results in terms of error score ($E \downarrow$), LLM-evaluated quality score ($S \uparrow$), and cost (relative to baseline, δ).

As shown in Table 14, the removal of any in-

Model Series	Size	Method	CNN/DailyMail	HANNA	TruthfulQA	Heuristic Gen.
Qwen2.5	32B	Implicit	15.80%	6.50%	28.25%	1.63%
		MARKERGEN	**16.43%**	**23.00%**	**36.91%**	**5.69%**
	14B	Implicit	9.40%	7.04%	**28.87%**	1.63%
		MARKERGEN	**11.60%**	**11.00%**	24.12%	**3.25%**
	7B	Implicit	4.20%	1.51%	23.24%	0.00%
		MARKERGEN	**14.80%**	**16.67%**	**25.62%**	**4.07%**
Llama3.1	70B	Implicit	7.40%	2.50%	**33.81%**	0.00%
		MARKERGEN	**9.02%**	**6.03%**	30.31%	**1.63%**
	8B	Implicit	**12.27%**	3.02%	19.90%	0.00%
		MARKERGEN	10.69%	**4.15%**	**20.52%**	0.00%

Table 9: Win rates (%) of Implicit baseline and MARKERGEN compared against Gemini-2.0 Flash using GPT-4o-mini as judge across four benchmarks. Higher win rates indicate better generation quality.

Model Series	Size	Method	S_4o	S_gemini	S_4omini	S_avg	σ^2
Qwen2.5	32B	Implicit	3.02	3.64	3.21	3.29	0.067
		MARKERGEN	**3.05**	**3.67**	**3.25**	**3.32**	0.066
	14B	Implicit	2.92	3.55	3.15	3.21	0.099
		MARKERGEN	**2.93**	**3.58**	**3.16**	**3.22**	0.092
	7B	Implicit	2.68	3.39	3.04	3.04	0.125
		MARKERGEN	**2.83**	**3.45**	**3.07**	**3.12**	0.092
Llama3.1	70B	Implicit	2.84	3.55	3.09	3.16	0.109
		MARKERGEN	**3.08**	**3.69**	**3.36**	**3.38**	0.068
	8B	Implicit	2.81	3.37	3.04	3.07	0.067
		MARKERGEN	2.78	**3.38**	**3.18**	**3.11**	0.074

Table 10: Evaluation scores of MARKERGEN and Implicit baseline on CNN/DailyMail benchmark using three judges (GPT-4o, Gemini-2.0 Flash, GPT-4o-mini).

dividual stage leads to a significant increase in error or a drop in quality. Notably, the two-stage variant (1+2) achieves the lowest error (2.66) with a relatively low cost (0.79δ), confirming its effectiveness and efficiency. Meanwhile, the full three-stage setup offers a balanced trade-off, delivering strong performance with moderate cost. These results reinforce the necessity of each stage and validate our full method design.

E.5 Residual Length Error Analysis in MARKERGEN

This subsection focuses on analyzing the residual length errors in the MARKERGEN framework. Building upon the sub-decomposition of LCTG errors presented in Section 2, we eliminate identifying and counting errors through Auxiliary Length Marker Insertion Decoding 3.1. Moreover, by employing the Three-Stage Decoupled Generation strategy 3.2, we effectively reduce aligning errors, thus improving the robustness of all models in semantic expansion under precise length modeling with explicit length markers. This approach ensures semantic integrity while enhancing text generation quality through a clearer, more in-depth analysis of LLMs LCTG sub-capabilities. Ultimately, residual LCTG errors are primarily driven by minimal aligning errors.

E.6 Cross-layer Attention Analysis from the MARKERGEN Perspective

In this section, we perform a cross-layer attention analysis from the MARKERGEN perspective. By examining attention patterns across different layers of the model, we aim to gain a better understanding of how length and semantic information are processed at various stages of generation, providing insights into improving the accuracy of LCTG tasks.

Combining the analyses from Figures 6, 10, 11, 12, and 13, we infer that in the shallow layers, attention is primarily focused on the length information represented by the length markers. This suggests that the models early stages prioritize processing and understanding the input length. The higher entropy in these layers indicates that the model needs to integrate various details and information to effectively comprehend the input. As the

Model Series	Size	Method	S_4o	S_gemini	S_4omini	S_avg	σ^2
Qwen2.5	32B	Implicit	3.20	3.73	3.67	3.53	0.056
		MARKERGEN	**3.29**	**3.75**	**3.72**	**3.59**	0.044
	14B	Implicit	3.15	3.61	3.55	3.44	0.041
		MARKERGEN	**3.16**	3.61	3.55	3.44	0.039
	7B	Implicit	3.09	3.31	3.47	3.29	0.024
		MARKERGEN	**3.12**	**3.36**	**3.50**	**3.33**	0.024
Llama3.1	70B	Implicit	3.18	3.66	3.61	3.48	0.047
		MARKERGEN	**3.25**	**3.74**	**3.63**	**3.54**	0.044
	8B	Implicit	3.13	3.45	3.54	3.37	0.030
		MARKERGEN	**3.19**	**3.50**	**3.60**	**3.43**	0.031

Table 11: Evaluation scores of MARKERGEN and Implicit baseline on HANNA benchmark using three judges (GPT-4o, Gemini-2.0 Flash, GPT-4o-mini).

Model Series	Size	Method	S_4o	S_gemini	S_4omini	S_avg	σ^2
Qwen2.5	32B	Implicit	4.56	3.94	4.45	4.32	0.073
		MARKERGEN	**4.63**	**4.09**	**4.54**	**4.42**	0.055
	14B	Implicit	4.47	**3.95**	**4.44**	**4.29**	0.056
		MARKERGEN	4.44	3.93	4.43	4.27	0.058
	7B	Implicit	4.24	3.72	4.29	4.09	0.066
		MARKERGEN	**4.34**	**3.83**	**4.33**	**4.17**	0.056
Llama3.1	70B	Implicit	4.53	4.09	4.46	4.36	0.038
		MARKERGEN	**4.59**	**4.14**	**4.48**	**4.40**	0.038
	8B	Implicit	**4.20**	3.62	4.22	4.01	0.076
		MARKERGEN	4.19	3.62	**4.25**	**4.02**	0.080

Table 12: Evaluation scores of MARKERGEN and Implicit baseline on TruthfulQA benchmark using three judges (GPT-4o, Gemini-2.0 Flash, GPT-4o-mini).

model progresses to deeper layers, attention shifts from the length information to the adjacent semantic content. The lower entropy in these layers indicates that the model refines its focus, extracting key features and generating more relevant output.

This pattern of attention distribution aligns with the findings from (Moon et al.), which emphasize that length modeling in the early layers serves as a foundation for semantic processing in the later layers. Our analysis further supports the notion that LCTG tasks depend on a dynamic interaction between length control and semantic generation, where early layers focus on length constraints and deeper layers prioritize semantic coherence.

F Prompt for Three-Stage Decoupled Generation

The following three prompts correspond to the three stages of our proposed method. We present here the actual prompt templates used in the TruthfulQA task as representative examples: Stage One for planning (Figure 14), Stage Two for semantic integrity (Figure 15), and Stage Three for length-constrained rewriting (Figure 16).

Model Series	Size	Method	S_4o	S_gemini	S_4omini	S_avg	σ^2
Qwen2.5	32B	Implicit	4.53	4.14	3.80	4.16	0.089
		MARKERGEN	**4.64**	**4.22**	**4.14**	**4.33**	0.049
	14B	Implicit	4.42	3.98	3.80	4.07	0.068
		MARKERGEN	**4.46**	**4.07**	**4.00**	**4.18**	0.040
	7B	Implicit	3.97	3.70	3.42	3.69	0.050
		MARKERGEN	**4.52**	**4.15**	**4.13**	**4.26**	0.031
Llama3.1	70B	Implicit	4.47	3.98	3.74	4.07	0.093
		MARKERGEN	**4.54**	**4.08**	**3.98**	**4.20**	0.058
	8B	Implicit	4.21	3.89	3.72	3.94	0.041
		MARKERGEN	**4.47**	**4.04**	**4.03**	**4.18**	0.042

Table 13: Evaluation scores of MARKERGEN and Implicit baseline on Heuristic Generation benchmark using three judges (GPT-4o, Gemini-2.0 Flash, GPT-4o-mini).

Variant	Components	Interval	E (↓)	S (↑)	Cost
Baseline (0)	Direct Generation	64	30.27	4.51	δ
w/o Tool (1)	Plan + Generation	64	27.58	4.60	1.05δ
		Decaying	20.44	4.58	1.56δ
w Tool (2)	Insertion Strategy	64	5.96	4.27	0.79δ
		Decaying	3.58	4.27	0.69δ
w/o Tool (w decouple) (3)	Decoupled Generation	64	26.11	4.53	1.20δ
		Decaying	22.87	4.52	1.11δ
Two Stage (1+2)	Plan + Insertion Strategy	64	3.93	4.32	0.84δ
		Decaying	**2.66**	4.28	0.79δ
w/o Tool (w plan & decouple) (1+3)	Plan + Decoupled Generation	64	25.89	**4.63**	1.56δ
		Decaying	23.19	4.60	1.51δ
w Tool (w/o plan & w decouple) (2+3)	Insertion + Decoupled Generation	Decaying	5.11	4.50	1.10δ
Three Stage (1+2+3)	Plan + Insertion + Decoupled Gen.	Decaying	4.48	4.54	1.46δ

Table 14: Extended ablation study analyzing the impact of different components combinations. δ: cost relative to baseline.

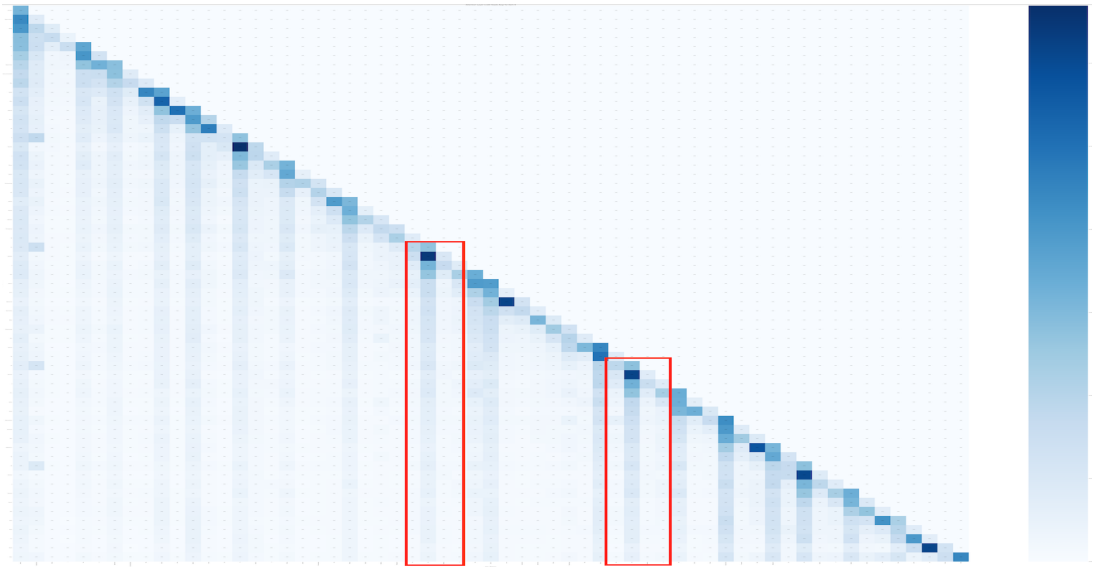


Figure 11: Attention Matrices of the first layers with Insertion Interval $n = 4$

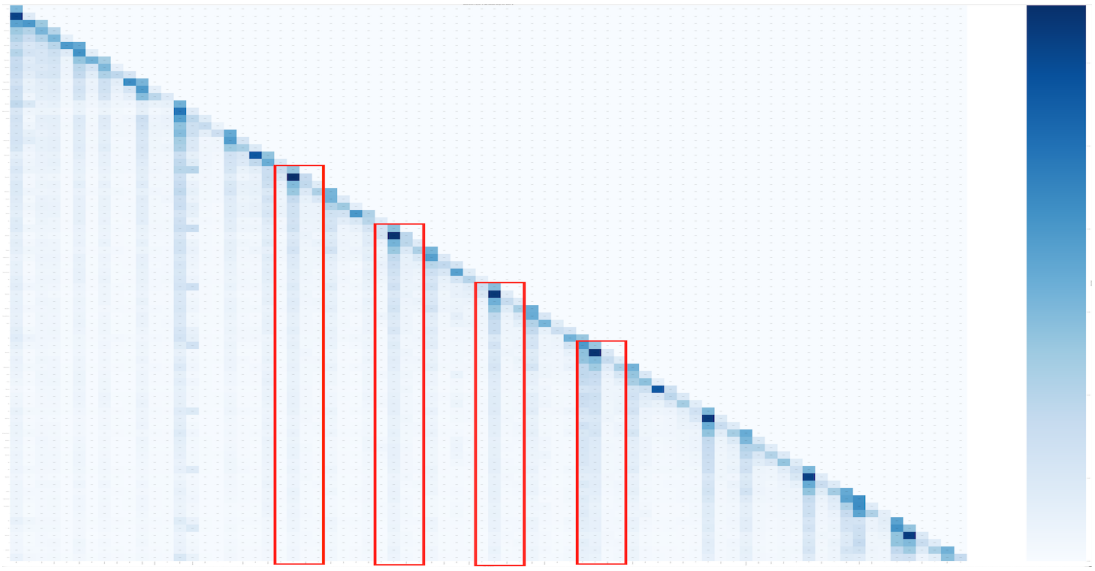


Figure 12: Attention Matrices of the first layers with Insertion Interval $n = 8$

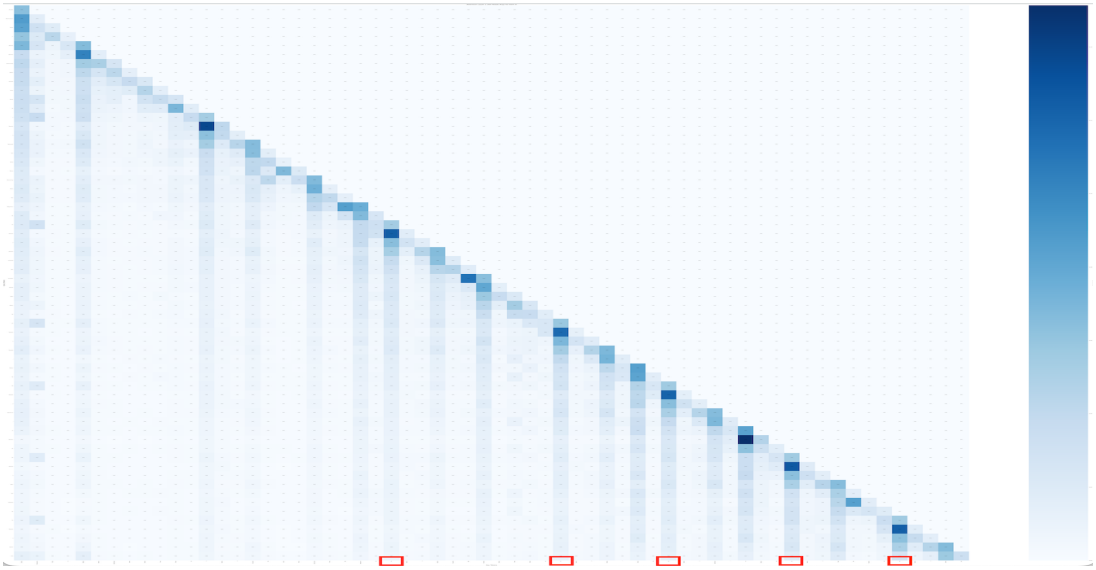


Figure 13: Attention Matrices of the first layers with Insertion Interval is Decaying

Stage One Prompt: Planning

Length Definition. A word is defined as any standalone word, number, or symbol, including punctuation and special symbols.
You are tasked with generating a high-quality and truthful answer to the following question, aiming for approximately `{target_length}` words.
Your answer must be factually accurate, free from any false information or hallucinations. Ensure that all statements can be supported by reliable sources.
Planning Task:

- Understand the question: Comprehend what is being asked.
- Research and gather facts: Collect accurate and relevant information.
- Organize the response: Structure the answer logically.
- Ensure completeness: Address all aspects of the question.
- Maintain accuracy and relevance: Avoid unnecessary digressions.

After planning, generate the answer based on this structure, maintaining logical consistency and factual accuracy.

Figure 14: Stage One Prompt: Planning

Stage Two Prompt: Semantic Integrity

Answer generation task:
Generate a comprehensive and precise answer to the question based on the plan, ensuring clarity, coherence, and factual accuracy.
The answer should be approximately `{target_length}` words in length.
A word is defined as any standalone word, number, or symbol, including punctuation and special symbols.
Only the answer text should be output; do not add any extra comments, notes, or explanations.
The answer should start with "`Answer generation task:`" and follow the format specified below.
Place "`###end`" at the absolute end of the answer to mark its completion.

`Question: "{prompt}"`

Figure 15: Stage Two Prompt: Ensuring Semantic Integrity

Stage Three Prompt: Length-Constrained Rewriting

Task Description:

1. In the first stage, we generated a high-quality answer without strict length control.
2. In this second stage, your task is to rewrite the high-quality answer to meet the specified length constraints.

Rewriting Requirements:

- Preserve core meaning, accuracy, and factual correctness.
- Match the target length of `{target_length}` words as closely as possible.
- Shorten or expand while maintaining clarity and integrity.
- Insert or remove detail as needed without altering facts.
- Output only the answer; no commentary or explanation.

Length Definition: A word is defined as any standalone word, number, or symbol, including punctuation and special symbols.
Length Feedback: The high-quality answer contains `{actual_length}` words. It `exceeds or falls short of` the target length by `{abs(length_difference)}` words.
Answer generation task:
After planning the adjustments, rewrite the answer in one go, adhering to the planned structure and word count.
Do not truncate unfinished sentences just to match the target.
Insert length markers during generation:
Start with larger intervals, then reduce spacing for detailed content. Markers should be numbered and evenly placed.
Example (Target length: 70 words):

```
Answer generation task: The golden light of the setting sun bathed the city streets in a
warm glow, [16 words] casting long shadows as people rushed home after a busy day. The
streets buzzed with [32 words] activity, cars honking, and pedestrians chatting. Amid the
hustle, a young couple [48 words] walked hand in hand, lost in conversation [56 words].
The sound of [60 words] their laughter mingled with [64 words] the noise [66 words] of
the [68 words] city [69 words]. [70 words] ###end
```

Final Prompt Input:

```
{task_description}
{rewrite_requirements}
{length_definition}
The Question is {prompt}
First stage High-quality answer: {generated_answer}
{length_feedback}
{answer generation task}
```

Figure 16: Stage Three Prompt: Length-Constrained Rewriting