GCoT-Decoding: Unlocking Deep Reasoning Paths for Universal Question Answering

Guanran Luo[1], Wentao Qiu[1], Zhongquan Jian[5], Meihong Wang[1], Qingqiang Wu[1,2,3,4,*]
[1]School of Informatics, Xiamen University [2]School of Film, Xiamen University
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, Xiamen University
[4]Institute of Artificial Intelligence, Xiamen University
[5]School of Computer and Data Science, Minjiang University
luoguanran@stu.xmu.edu.cn, wuqq@xmu.edu.cn

Abstract

Chain-of-Thought (CoT) reasoning can enhance large language models (LLMs), but it requires manually designed prompts to guide the model. Recently proposed CoT-decoding enables the model to generate CoT-style reasoning paths without prompts, but it is only applicable to problems with fixed answer sets. To address this limitation, we propose a general decoding strategy—GCoT-decoding—that extends applicability to a broader range of question-answering tasks. GCoT-decoding employs a two-stage branching method combining Fibonacci sampling and heuristic error backtracking to generate candidate decoding paths. It then splits each path into a reasoning span and an answer span to accurately compute path confidence, and finally aggregates semantically similar paths to identify a consensus answer, replacing traditional majority voting. We conduct extensive experiments on six datasets covering both fixed and free QA tasks. Our method not only maintains strong performance on fixed QA but also achieves significant improvements on free QA, demonstrating its generality.

1 Introduction

Chain-of-thought (CoT) prompting is a simple but powerful way to elicit multi-step reasoning from large language models (LLMs), and can substantially improve benchmark performance (Kojima et al., 2022; Wei et al., 2022; Yao, 2024; Yasunaga et al., 2023; Zhou et al., 2022a; Lightman et al., 2023; Uesato et al., 2022; Xie et al., 2023; Golovneva et al., 2023). Most prior work, however, operates at the *prompt* level: carefully engineered instructions and exemplars are used to steer models towards explicit CoT traces. Such prompts inherit the designer's biases and often need to be re-tuned across tasks and output formats (Wang et al., 2022b; Ye and Durrett, 2022; Zhou et al., 2022b; Ge et al., 2025). A complementary line of work instead modifies the *decoding* process, for example via self-consistency (Wang et al., 2022a), contrastive decoding (Li et al., 2022), or context-aware decoding (Shi et al., 2024), but these methods typically rely on extra signals and still assume relatively rigid answer formats.

Table 1: Comparison of CoT-decoding in free-form vs. fixed-format QA tasks.

	Free QA	**Fixed QA**
Example	**Q**: What do Woodrow Wilson, George W. Bush, and James Monroe have in common? **k=1**: They all served as **presidents of the United States**. **k=2** : They were all **American leaders** involved in major wars. **k=3**: Each of them occupied the White House as **U.S. president**.	**Q**: A factory makes 3 toys per hour. How many toys after 8 hours? **k=1**: $3 \times 8 = \mathbf{24}$ (0.93) **k=2**: 3 times 8 is **24** (0.91) **k=3**: = **24** (0.85)
Answer Space	∞	$\mathbb{N}$
Exact Span Match	×	✓
Majority Vote Aggregation	×	✓

This motivates a natural question: *Can we explore and select CoT reasoning paths purely from the geometry of the base model's decoding process, without task-specific prompts or fixed answer formats?* CoT-decoding (Wang and Zhou, 2024) is an important first step. It perturbs the first decoding step by sampling the top-k alternatives, greedily rolls out a CoT trace from each seed, extracts an answer span, and uses the average top-1 vs. top-2 logit gap on that span as a path-level confidence score. Paths that lead to the same span are aggregated, and the answer with the highest cumulative confidence is returned.

While effective on fixed-format QA, this procedure hinges on two assumptions: (i) that a canonical answer span can be extracted reliably, and (ii) that branching only at the first decoding step and

*Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2026, pages 6398–6414
July 2-7, 2026

exploring seeds in index order is sufficient. Table 1 illustrates both limitations. On a fixed-answer question (right), all high-likelihood CoT paths end with the same numeric span "24", so exact span matching and majority voting are straightforward. On a free-form QA question (left), the correct answer is phrased in several semantically equivalent ways, and another path mentions a plausible but wrong alternative; there is no unique span for aggregation by exact match. Moreover, as our empirical analysis shows (see Appendix A), early high-probability seeds often form clusters of very similar yet incorrect continuations, while correct CoT paths are buried deeper in the ranked list.

In this paper, we revisit CoT-decoding from a decoding-time perspective and organize the design space into three questions: (1) *Exploration*: how to spend a small path budget on diverse but plausible reasoning directions rather than near-duplicate early seeds? (2) *Confidence*: how to score paths without relying on a single, task-specific answer span, in both fixed-answer and free-form settings? (3) *Aggregation*: how to robustly pool free-form answers so that tiny logit differences between nearly equivalent paths do not cause unstable predictions?

To answer these questions, we propose General Chain-of-Thought Decoding (**GCoT-decoding**), a modular three-layer decoding framework. Our design is motivated by a simple observation: in decoding-time reasoning, the key challenge is not merely to generate more paths, but to allocate a limited path budget to *non-redundant* and *repairable* regions of the search space. Accordingly, an effective decoding strategy should satisfy three properties: (i) it should cover the rank axis broadly enough to avoid wasting the budget on near-duplicate early candidates; (ii) it should allocate additional computation only at positions where the current trajectory shows clear signs of failure; and (iii) it should aggregate semantically equivalent free-form answers without relying on exact string overlap.

GCoT-decoding is designed exactly around these requirements. At the exploration layer, GCoT combines Fibonacci-based seeding with a single local-minimum backtracking step to allocate a small path budget to both diverse global starts and locally failing regions of the decoding trajectory. At the confidence layer, it views each path as a "reasoning trace + answer continuation" and assigns a length-aware top-2 logit gap score, with an optional LCS-based SpanAlign variant that focuses on the aligned answer segment. At the aggregation layer, GCoT applies greedy semantic clustering over answer strings and selects the representative from the cluster with the highest accumulated confidence, aggregating paraphrase-equivalent paths without relying on a fixed answer format.

We evaluate GCoT-decoding on six datasets spanning fixed-answer and free-form QA. GCoT matches or slightly improves over standard multi-path decoders on fixed-answer tasks, and yields consistent gains on free-form benchmarks where span-based CoT-decoding struggles. It also composes cleanly with few-shot CoT prompting and reasoning-tuned models, providing additional improvements on top of strong baselines. Ablation studies isolate the role of each layer and show that Fibonacci-based multi-path exploration together with greedy semantic clustering accounts for most of the gains, with local-minima backtracking providing a smaller but consistent refinement. We support these design choices with an empirical analysis of span sensitivity and exploration failure modes for CoT-decoding, which we report in Appendix A.

Overall, our contributions are:

- **A general decoding framework for universal QA.** We propose GCoT-decoding, a prompt-free multi-path decoding framework that removes the fixed-answer-span assumption in prior CoT-decoding and extends applicability from fixed-format QA to free-form QA.

- **A better path-budget allocation strategy.** We introduce a two-stage exploration mechanism that combines Fibonacci-seeded global branching with local confidence-valley backtracking. The first component spreads a small seed budget over the ranked candidate axis to avoid redundant early clusters, while the second selectively repairs trajectories that show early signs of failure.

- **A lightweight aggregation mechanism for free-form answers.** We replace exact-span voting with greedy semantic clustering, which aggregates paraphrase-equivalent answers at low computational cost and delivers most of the gain of stronger aggregation methods without their latency overhead.

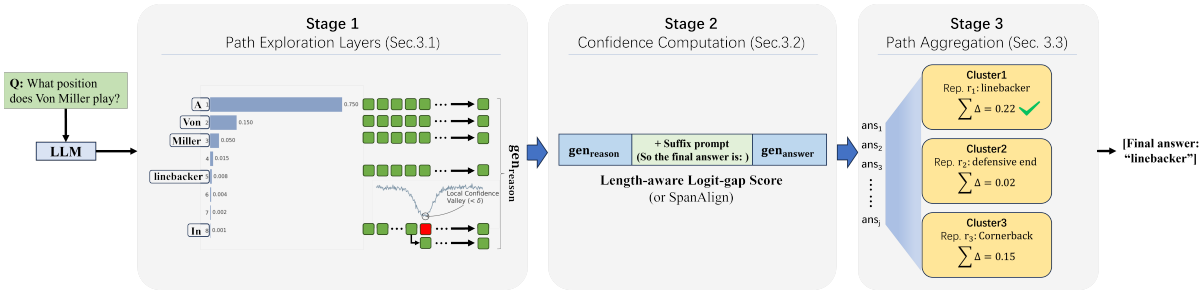


Figure 1: Overall workflow of GCoT-decoding. A small number of reasoning paths are first generated via Fibonacci seeding and confidence-based backtracking, then scored by length-aware logit gaps, and finally aggregated through greedy semantic clustering over answer strings.

2 Related Work

Chain-of-Thought. Chain-of-Thought (CoT) prompting decomposes complex tasks into intermediate reasoning steps and has inspired a series of automated and structured extensions, including Auto-CoT (Zhang et al., 2022), Synthetic Prompting (Shao et al., 2023), Contrastive Denoising CoT, Faithful CoT, and KG-COT, which aim to improve generation quality and logical fidelity (Wei et al., 2022; Kojima et al., 2022; Zhang et al., 2022; Shao et al., 2023; Zhou et al., 2024; Lyu et al., 2023; Zhao et al., 2024; Sun et al., 2026). Self-Consistency further enhances performance by aggregating diverse reasoning paths (Wang et al., 2022a; Wang and Zhou, 2024; Ji et al., 2026; Ge et al., 2026). However, most prompting-based methods rely heavily on labeled examples, handcrafted templates, or predefined outputs, limiting scalability. In contrast, our GCoT-Decoding removes these dependencies to enable broader applicability.

Prompting Methods to Enhance Reasoning. Efforts to improve prompting strategies include paraphrasing, active example selection, analogical cues, and instruction tuning (Chen et al., 2024; Diao et al., 2023; Yasunaga et al., 2023; Zhang et al., 2024b; Ho et al., 2022; Liu and Henao, 2025). Recent work also explores context-aware decoding and weakly-supervised aggregation to improve robustness (Shi et al., 2024; Ling et al., 2023; Arora et al., 2022), though such methods often introduce additional annotation or computation costs. Prompt sensitivity and task specificity remain common bottlenecks.

Decoding Strategies to Enhance Reasoning. Beyond prompting, decoding-time strategies provide an alternative route for eliciting reasoning. Early contrastive decoding diversified outputs without relying on prompts (Li et al., 2022; Yao, 2024), while self-evaluation, confidence-based scoring, and preference-guided optimization have been proposed to refine multi-step reasoning (Xie et al., 2023; Wang et al., 2024; Taubenfeld et al., 2025; Zhang et al., 2024a). Tree-of-Thoughts (Yao et al., 2023) and CoT-decoding (Wang and Zhou, 2024) treat reasoning as a structured exploration process, with the latter showing that top-k sampling alone can reveal rich reasoning paths. Speculative decoding methods improve efficiency but are less focused on reasoning quality (Chen et al., 2025; Xu et al., 2025). A recent survey by (Welleck et al., 2024) provides a comprehensive overview of decoding strategies for reasoning tasks.

3 Method

Given a question x and a language model $p_\theta(\mathbf{y} \mid x)$, our goal is to uncover correct but non-prominent chain-of-thought (CoT) reasoning paths without assuming a fixed answer format. We organize GCoT-decoding as a *three-layer* process: (i) a coarse path exploration layer that seeds a small number of diverse reasoning directions; (ii) a local error-repair layer that backtracks from confidence valleys; and (iii) an answer aggregation layer that pools evidence across paraphrase-equivalent answers.

Figure 1 summarizes the workflow. Sec. 3.1 describes the branching strategy for constructing candidate paths, Sec. 3.2 defines our length-aware confidence scores and the SpanAlign variant, and Sec. 3.3 introduces greedy semantic clustering for path aggregation.

3.1 Two-stage branching for path exploration

Our analysis in Appendix A.2 shows that, under standard CoT-decoding, the ranked candidate list in the first few decoding steps often collapses into

clusters of very similar but incorrect reasoning trajectories. Spending a small path budget on the first few indices therefore risks exploring many near-duplicates of the same erroneous hypothesis.

GCoT-decoding instead adopts a two-stage branching scheme: a global layer that performs *one-step diversification + greedy rollout* using Fibonacci indices, and a local layer that performs *backtracking from the first confidence minimum* along each path.

Layer 1: Fibonacci-seeded greedy rollouts. Let $\tilde{y}_1^{(1)}, \tilde{y}_1^{(2)}, \ldots$ denote candidate tokens at the first decoding step, sorted by $p_\theta(y_1 \mid x)$ in descending order. Rather than taking the first K candidates, we choose indices from the Fibonacci sequence

$$S_{\text{fib}} = \{F_1, F_2, \ldots, F_K\}, \qquad F_n = F_{n-1} + F_{n-2},\ F_1 = 1,\ F_2 = 2 \tag{1}$$

and initialize K reasoning seeds $\tilde{y}_1^{(F_1)}, \ldots, \tilde{y}_1^{(F_K)}$. For each seed, the remaining tokens are generated by greedy decoding, yielding a set of candidate CoT paths $\{p_k\}_{k=1}^{K}$, where $p_k = (y_{k,1}, \ldots, y_{k,T_k})$.

Fibonacci indices implement a roughly log-spaced coverage of the rank axis. This design is not merely based on a simple "logarithmic coverage" intuition, but rather on a quantitative observation of the distribution characteristics of error paths in the decoding space. As shown in Appendix A.2, when the top-ranked path is incorrect, the immediately following candidate paths have a high probability of repeating similar errors, whereas correct paths appear significantly more frequently at larger indices. The design intent of Fibonacci sampling is precisely to utilize its non-linear step size to rapidly skip over redundant error path clusters in the high-probability head region, thereby allocating the limited path budget to more diverse and potentially correct regions. This logic is validated in our ablation study (Table 4), where replacing Fibonacci sampling with traditional top-k sampling under a fixed budget causes accuracy to drop drastically.

Layer 2: backtracking from local confidence minima. Even with diverse seeds, a greedy rollout can drift toward an incorrect answer. Empirically, such drifts are accompanied by sharp local drops in token-level confidence along the path. We treat these as *first-hitting times* of a low-confidence region and use them as signals for local error repair.

For a greedy path $p_k = (y_{k,1}, \ldots, y_{k,T_k})$, define the token-level confidence

$$s_{k,t} = p_\theta(y_{k,t} \mid x, y_{k,<t}), \qquad t = 1, \ldots, T_k. \tag{2}$$

Starting from $t = 3$, we collect indices that are strict local minima below a threshold δ:

$$S_k = \{\, t \mid 3 \le t \le T_k,\ s_{k,t} < s_{k,t-1},\ (t < T_k \Rightarrow s_{k,t} < s_{k,t+1}),\ s_{k,t} < \delta \,\}, \tag{3}$$

and define the backtracking index

$$b_k = \begin{cases} \min S_k, & S_k \neq \varnothing, \\ -1, & S_k = \varnothing. \end{cases} \tag{4}$$

If $b_k \neq -1$, we step back to y_{k,b_k-1} and re-branch on K' alternatives (again using Fibonacci indices over the candidate list at that position),

$$\mathbf{y}_{k,<b_k}^{(m)} = (y_{k,1}, \ldots, y_{k,b_k-2}, y_{k,b_k-1}^{(m)}), \qquad m \in \{F_1, \ldots, F_{K'}\}, \tag{5}$$

and complete each prefix with greedy decoding to obtain new paths $\{p_{k,m}\}_{m=1}^{K'}$. If $S_k = \varnothing$, we simply keep p_k.

This design has two effects. First, it focuses additional computation precisely at early confidence valleys, where the probability trajectory $(s_{k,t})_t$ indicates that the current semantic direction has fallen off the model's high-confidence manifold. Second, it avoids perturbing every token position. Pseudocode for the two-stage branching scheme is provided in Appendix B.

3.2 Length-aware logit-gap confidence

After generating a set of candidate reasoning paths, we must assign a scalar confidence score to each path for later aggregation. Standard CoT-decoding for fixed-answer tasks often uses the average difference between the top-1 and top-2 logits over the answer span as a confidence proxy (Wang and Zhou, 2024), but this requires a pre-defined answer span and is brittle when answer formats vary.

GCoT-decoding replaces this with a two-step scoring scheme: (i) a *length-aware* logit-gap score based on an explicit split between reasoning and answer segments, and (ii) an optional *SpanAlign* variant that further focuses on tokens aligned between the reasoning and answer continuations.

Splitting reasoning and answer segments. For each path p_k, we first let the model produce a full CoT reasoning segment $\text{gen}_{1,k}$. We then append a short continuation prompt such as *"So the answer is:"* and decode a concise answer segment

$gen_{2,k}$ (defined for each path k as the explicit answer continuation). This template is used purely as a *post-hoc answer extractor* after the reasoning is complete and does not affect how the CoT itself is generated; in Appendix G we show that replacing it with semantically equivalent phrases leads to only minor variation in accuracy.

CoT paths that explore deeper reasoning directions tend to have longer $\text{gen}_{1,k}$ and more coherent answer segments. Motivated by this, we define the base confidence of path k as

$$\Delta_{k,\text{answer}}^{\text{GCoT-decoding}} = \underbrace{\frac{\log\big(1+|\text{gen}_{1,k}|\big)}{\max_{i\in\{1,\dots,K\}}\log\big(1+|\text{gen}_{1,i}|\big)}}_{\text{length normalization over reasoning segments}} \times \underbrace{\frac{1}{|\text{gen}_{2,k}|}\sum_{a_t\in\text{gen}_{2,k}}\big(p(a_t^1)-p(a_t^2)\big)}_{\text{average top-2 logit gap over the answer segment}} \tag{6}$$

where $p(a_t^1)$ and $p(a_t^2)$ denote the probabilities of the top-2 tokens at decoding step t in the answer segment $gen_{2,k}$. The first factor encourages paths with sufficiently long reasoning, while the second captures how confidently the model commits to the final answer.

Comparative Variant: SpanAlign via LCS We introduce SpanAlign as a *comparative variant*. We designed SpanAlign specifically to demonstrate that relying on exact span alignment is fundamentally too rigid for open-ended, free-form QA tasks. SpanAlign relies on Longest Common Subsequence (LCS) matching between the intermediate reasoning path $gen_{1,k}$ and the final answer continuation $gen_{2,k}$. Before computing the LCS, we normalize both strings by lowercasing and stripping pure punctuation tokens. Let $LCS(gen_{1,k}, gen_{2,k}) = (s_{1,1}, ..., s_{1,m}; s_{2,1}, ..., s_{2,n})$ be the aligned subsequences, with total length L. We focus on the terminal aligned spans $s_{1,m}$ and $s_{2,n}$, and define:

$$\Delta_{k,\text{answer}}^{\text{GCoT+SpanAlign}} = \frac{1}{L}\left(\sum_{a_{1,t}\in a_{1,m}}\big(p(a_{1,t}^1)-p(a_{1,t}^2)\big) + \sum_{a_{2,t}\in a_{2,n}}\big(p(x_{a,t}^1)-p(a_{2,t}^2)\big)\right) \tag{7}$$

For the vast majority of open-ended, free-form, and general reasoning tasks, we recommend keeping SpanAlign **disabled** by default, relying instead on our core Greedy Semantic Clustering. SpanAlign should only be enabled as an optional feature when handling highly constrained tasks that require verbatim extractive matching.

3.3 Greedy semantic clustering for path aggregation

If we were to select the final prediction solely by $\max_k \Delta_{k,\text{answer}}$, small perturbations in logits could cause large jumps in the chosen path, especially when several answers are close in confidence. Aggregating across multiple paths can mitigate this sensitivity, but standard majority voting is not applicable on open-ended tasks where answers are free-form text and exact string matches are rare. We therefore aggregate at the level of *semantic answer clusters*.

Let $\{p_i\}_{i=1}^K$ denote the candidate paths produced by the branching stages, with final answers $g_i = \text{gen}_{2,i}$ and confidence scores $c_i = \Delta_{i,\text{answer}}$. We maintain a set of semantic groups $\{G_j\}_{j=1}^N$ with representative answers $\{r_j\}_{j=1}^N$, initially empty. For each answer g_i in index order, we compute cosine similarities

$$s_{i,j} = \cos\big(\phi(g_i),\, \phi(r_j)\big), \qquad j = 1, \dots, N, \tag{8}$$

where $\phi(\cdot)$ is a sentence embedding function. We then assign g_i according to the greedy rule

$$j^* = \begin{cases} \min\{\, j \in \{1,\dots,N\} \mid s_{i,j} \ge \tau\}, & \text{if } \max_{1\le j\le N} s_{i,j} \ge \tau, \\ N+1, & \text{otherwise,} \end{cases} \tag{9}$$

which always chooses the first existing cluster above a similarity threshold τ, or creates a new cluster if none qualify. We update

$$G_{j^*} \leftarrow G_{j^*} \cup \{g_i\}, \qquad r_{j^*} = \begin{cases} r_{j^*}, & j^* \le N, \\ g_i, & j^* = N+1, \end{cases} \qquad N \leftarrow \max(N, j^*). \tag{10}$$

After all K answers are processed, we compute the cumulative confidence of each group

$$C_j = \sum_{g_i \in G_j} c_i, \qquad j = 1, \dots, N, \tag{11}$$

and select the representative $r_{j_{\max}}$ with $j_{\max} = \arg\max_j C_j$ as the final output.

We further ablate the sentence embedding model and find that performance varies only slightly (Appendix H), suggesting that the clustering module is relatively insensitive to the particular off-the-shelf encoder. Pseudocode for the aggregation procedure is also given in Appendix B.

	Spec Ans	GSM8K			MultiArith			Sports understanding		
		Mistral-7B	Gemma-7B	Llama-3.1-8B	Mistral-7B	Gemma-7B	Llama-3.1-8B	Mistral-7B	Gemma-7B	Llama-3.1-8B
Greedy	×	10.5	11.6	17.9	16.0	18.7	38.8	49.6	61.2	51.6
Temperature sampling	×	8.4	7.9	13.1	15.2	18.8	36.2	48.9	60.1	52.4†
Top-k sampling	×	5.1	6.2	14.2	13.3	17.3	37.0	50.3	58.0	51.9
Beam search	×	6.7	10.2	17.1	15.5	17.9	38.1	48.2	59.9	50.7
CoT-decoding	✓	21.9♠	25.4♠	36.3†	40.6♠	43.8♠	72.3†	50.6	68.4♠	51.0
Self-consistency	✓	16.3	17.2	28.5	21.7	22.9	46.9	52.9♠	63.9	54.6†
GCoT-decoding + SpanAlign	×	10.7	15.4	34.0	16.8	19.7	69.3	48.0	67.2†	52.0
GCoT-decoding	×	18.0†	21.8†	41.7♠	31.3†	22.8†	74.3♠	52.0†	65.2	58.0♠

Table 2: Accuracy comparison of decoding strategies on fixed QA tasks; the top-ranked is marked with ♠ and the second-ranked is marked with †. Spec Ans indicates whether the decoding strategy relies on specific answer spans. The top section lists single-path decoding strategies; the bottom section shows multi-path decoding strategies.

	SQuAD v1.1 (contextual)						BARQA (contextual)						Auto categorization (context-free)					
	Gemma-7B		Llama-3.1-8B		Qwen2.5-14B		Gemma-7B		Llama-3.1-8B		Qwen2.5-14B		Gemma-7B		Llama-3.1-8B		Qwen2.5-14B	
	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH	BLEU	MATCH
Greedy	3.3	42.8	8.3	60.6	21.4	67.2	4.7	36.6†	10.8	39.7	10.7†	44.4♠	5.8	16.8	5.1†	16.0†	8.5	29.0
Temperature sampling	3.1	40.1	7.5	57.2	17.1	64.1	4.5	32.1	7.3	37.4	7.7	42.5	6.0	13.6	4.9	13.3	6.6	27.9
Top-k sampling	2.8	35.2	5.4	51.0	13.1	55.1	2.9	33.3	6.8	37.2	6.4	40.0	4.3	13.7	4.5	11.2	5.6	26.0
Beam Search	3.2	41.9	7.9	59.3	20.0	66.0	4.2	35.4	10.0	38.5	10.1	42.1	5.3	15.0	4.7	15.4	8.1	28.4
CoT-decoding + Prompt-based	0.2	25.7	1.3	40.9	5.8	50.3	0.7	21.5	2.4	25.1	1.4	32.0	1.2	20.1	2.0	15.7	8.0	29.0
Self-consistency + Prompt-based	4.2†	36.7	3.2	43.2	12.1	58.0	2.2	26.1	3.6	30.4	1.5	33.5	7.4	20.3	3.1	14.1	5.3	29.8
GCoT-decoding + SpanAlign	3.9	48.9†	9.2†	62.0†	21.5†	69.6†	5.8†	36.5	10.9†	41.5†	10.9♠	43.3†	8.8†	23.3†	4.5	14.7	8.8†	30.2†
GCoT-decoding	4.9♠	54.6♠	10.0♠	67.2♠	23.2♠	71.4♠	10.9♠	37.7♠	12.3♠	44.1♠	10.2	38.9	8.9♠	24.6♠	6.8♠	20.0♠	10.6♠	30.5♠

Table 3: Performance of different models on free QA tasks; the top-ranked is marked with ♠ and the second-ranked is marked with †. The top section lists single-path decoding strategies; the bottom section shows multi-path decoding strategies.

4 Results and Analysis

4.1 Experimental setup

Datasets. We evaluate models on two categories of QA tasks: (1) *Fixed QA*, where the answer set or format is constrained, including **GSM8K** and **MultiArith** (Cobbe et al., 2021; Roy and Roth, 2015) for multi-step arithmetic reasoning, and **Sports understanding** (Suzgun et al., 2022) from Big-Bench-Hard for binary reasoning over sports-related sentences; and (2) *Free QA*, which involves open-ended or paragraph-level outputs, such as **SQuAD v1.1** (Rajpurkar et al., 2016) for extractive reading comprehension, **BARQA** (Srivastava et al., 2022) for context-dependent anaphora resolution, and **Auto Categorization** (Srivastava et al., 2022) for identifying semantic categories among object sets.

Baseline Methods and Evaluation Metrics. We primarily compare decoding-based methods, including **single-path sampling** strategies such as greedy decoding, temperature sampling ($t = 0.7$), and top-k sampling ($k = 10$); as well as **multi-path sampling** methods like beam search ($b = 10$), self-consistency ($k = 10$) (Wang et al., 2022a) and CoT-decoding (Wang and Zhou, 2024).

We do not include prompt-based methods as baselines, as they are orthogonal to GCoT-decoding and can be freely combined (see Section 4.3 for discussion). For **fixed QA**, we use *accuracy*, computed by comparing the extracted answer token against the ground truth—note this extraction is used only for evaluation, not confidence computation. For **free QA**, we evaluate with *BLEU* (Papineni et al., 2002) and *MATCH*, which checks whether the ground-truth span appears in the response. For GCoT-decoding variants, *BLEU* is calculated only on the final answer gen_2.

Model and Parameter Settings. In the main experiments, we evaluate four models: Mistral-7B (Jiang, 2024), Gemma-7B (Team et al., 2024), Llama3.1-8B (Grattafiori et al., 2024), and Qwen2.5-14B (Yang et al., 2024). For the model-scale ablation, we use the Qwen2.5 series at 3B, 7B, 14B, and 32B scales. We use all-MiniLM-L6-v2 (Reimers and Gurevych, 2019) as the embedding model. We set the first-stage branching number $k = 10$ and second-stage branching number $k' = 2$, branch only when confidence falls below a threshold δ of 0.2. During semantic aggregation of paths, we use a similarity threshold τ of 0.8. To ensure the stability and reliability of our findings, all results reported for the main experiments are calculated as the average of three independent runs.

4.2 Main results

Fixed QA. As shown in Table 2, GCoT-decoding outperforms all single-path decoding strategies

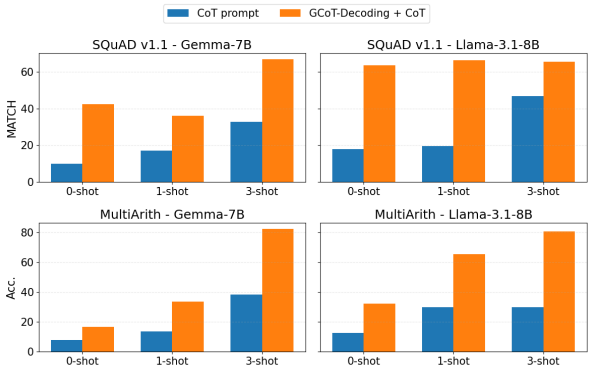


Figure 2: The results of combining GCoT-decoding with CoT prompting.

(greedy and sampling methods) and most multi-path decoding strategies (beam search and self-consistency) across all models and datasets. Although CoT-decoding achieves the highest accuracy on math reasoning tasks, its performance heavily relies on specific answer spans. This dependency explains its advantage in fixed QA tasks but also becomes a major bottleneck when extending to free QA tasks. In contrast, GCoT-decoding offers a more stable alternative that does not rely on answer spans, achieving competitive performance on fixed QA while delivering significant gains on free QA.

Free QA. As shown in Table 3, GCoT-decoding achieves the highest BLEU and MATCH scores in nearly all settings, significantly outperforming other methods in both generation quality and answer alignment. Even compared to variants such as CoT-decoding + Prompt-based and Self-consistency + Prompt-based, GCoT-decoding remains the top performer. In contrast, GCoT-decoding + SpanAlign suffers from performance drops due to frequent misalignment with incorrect spans. Overall, GCoT-decoding demonstrates stronger robustness and generality when tackling complex, free-form reasoning tasks.

4.3 Compatibility of GCoT-decoding with Prompting Methods

Although GCoT-decoding is a prompt-free method, this does not preclude its combination with prompt-based approaches; in fact, they are highly compatible. Experiments on MultiArith and SQuAD v1.1 using Gemma-7B and Llama-3.1-8B show (Figure 2) that merging GCoT-decoding with CoT prompting yields steady performance improvements across all few-shot settings in both fixed and free QA, with absolute gains of 10%–50%. This demonstrates that GCoT-decoding and CoT prompting synergize effectively, significantly enhancing LLM reasoning quality in few-shot scenarios. We provide the few-shot examples used in Appendix D.

Variant	GSM8K (Gemma-7B)	GSM8K (Mistral-7B)	SQuAD v1.1 (Gemma-7B)	SQuAD v1.1 (Llama-3.1-8B)
Fibonacci + greedy (ours)	21.8	18.0	54.6	67.2
top-k sampling (k=10)	7.9	6.2	42.1	50.4
top-p sampling (p=0.9)	8.6	7.0	43.5	51.3
temperature sampling (T=0.7)	9.4	7.8	45.0	52.6

Table 4: Ablation of path-generation strategies under a fixed budget of K=10 paths. All variants share the same backtracking and aggregation modules.

4.4 Ablation study

We ablate GCoT-decoding along its three main stages: (i) the path generation strategy, (ii) the backtracking rule, and (iii) the path aggregation module. Appendix C reports additional ablations, including alternative confidence computation schemes and multi-factor variants where several modules are simplified simultaneously.

Effect of path generation strategy. Our goal differs from generic diversity generation: instead of injecting randomness at every step, we only diversify the first token to open a few alternative reasoning directions and then greedily roll out each path. Fibonacci indices further spread this first-step sampling budget along the ranked candidates in a roughly log-spaced manner, avoiding redundant exploration of tightly clustered early hypotheses. Under a fixed budget of K=10 paths, Table 4 compares this Fibonacci-based scheme to standard step-wise stochastic sampling while keeping backtracking and aggregation fixed, and shows that replacing our "one-step diversification + greedy rollout" with top-k/top-p/temperature sampling drives GSM8K accuracy down to about 8–10% and reduces SQuAD v1.1 MATCH by 10–20 points.

Reliability of local-minima backtracking. We assess reliability by measuring trigger frequency and success rate on SQuAD v1.1 (Table 5). Local-minima backtracking is triggered on only about 28% of questions, yet fixes an otherwise wrong greedy answer in 36.5% of those cases, raising MATCH from 52.7 to 54.6. Random and late backtracking are always triggered but slightly underperform the no-backtracking baseline and have much lower conditional success rates (around 18–21%),

Variant	Backtracking trigger rate (%)	Success rate given trigger (%)	MATCH	BLEU
No-backtracking	–	–	52.7	8.7
Random backtracking	100.0	18.1	52.0	8.6
Late backtracking	100.0	20.4	51.8	8.5
Local-minima backtracking (ours)	28.0	36.5	54.6	9.1

Table 5: Backtracking variants on SQuAD v1.1 dev (Gemma-7B); "Success rate given trigger" is the fraction of triggered cases corrected by backtracking.

Aggregation variant	Extra time per question (sec.)	GSM8K Acc. (Gemma-7B)	SQuAD MATCH (Gemma-7B)
MaxPath (no aggregation)	0.0	15.3	41.9
Greedy clustering (ours)	0.2	21.8	54.6
LLM-based aggregation	8.3	22.1	55.8

Table 6: MaxPath vs. greedy semantic clustering and an LLM-based aggregation module (Gemma-7B). Extra time is measured relative to greedy decoding.

indicating that naive perturbations are not helpful. We further study the effect of allowing more backtracking points per path in Appendix F.

Greedy semantic clustering vs. LLM-based aggregation. We first compare GCoT-decoding with a MaxPath baseline that simply selects the single highest-confidence path: as shown in Table 6, greedy semantic clustering improves GSM8K accuracy from 15.3 to 21.8 and SQuAD MATCH from 41.9 to 54.6, with only 0.2 seconds of extra time per question. An LLM-based aggregator yields slightly higher scores than greedy clustering but incurs about 8.3 seconds of additional latency and is sensitive to the aggregation prompt. Our greedy clustering therefore offers most of the aggregation benefit over MaxPath at a fraction of the compute cost, matching our goal of a lightweight, robust aggregation module.

4.5 Quantitative and qualitative analysis

Quantitative Analysis. As shown in Figure 3(b), performance improves with scale, especially from 3B to 7B, with smaller gains beyond. *GCoT-decoding* consistently outperforms *+SpanAlign* across scales and shows greater robustness to domain shifts. Furthermore, Figure 3(a) demonstrates that increasing the number of decoding paths k initially improves performance but saturates after $k > 5$, with *GCoT-decoding* maintaining stronger and more stable gains than *+SpanAlign* across all k settings.

To fully understand the practical deployment cost, we evaluate the computational overhead of our proposed aggregation mechanisms. As detailed in Table 8, in our default setup (generating $k = 10$ paths), the base generation time typically ranges from 20 to 70 seconds per question. Our proposed greedy semantic clustering adds an extremely small overhead of just 0.2 seconds ($\sim 0.4\%$). This completely avoids the massive latency penalty of LLM-based aggregation (which adds 8.3 seconds, or $\sim 18.4\%$ overhead) while achieving comparable performance.

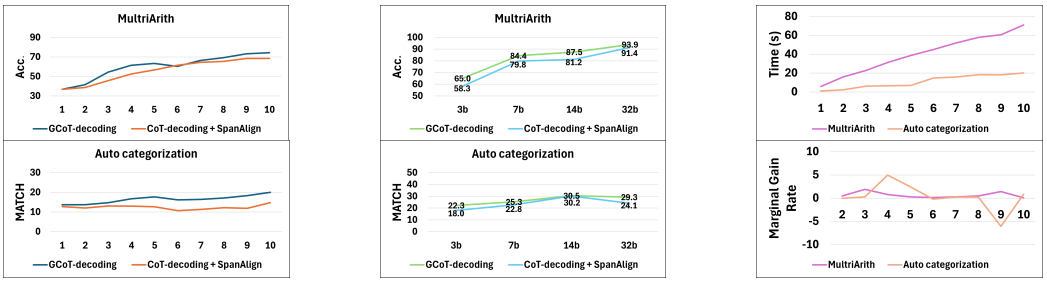
Finally, evaluating the overall trade-off, Figure 3(c) illustrates that the total time cost grows roughly linearly with k, while both tasks exhibit diminishing marginal gains. Combining these observations, the optimal "elbow" lies in the range of $k = 3 \sim 5$, where the marginal gain rate peaks and the computational time cost remains moderate.

How Fibonacci sampling works. Table 7 shows two case studies of Fibonacci sampling. In the war classification example, paths $k = 1$–3 all converge on related but wrong labels such as "diplomatic initiatives," and the correct label "historical wars" only emerges at $k = 8$, with a clear reasoning chain—illustrating how early high-probability seeds can cluster around the same mistake. Here, Fibonacci sampling skips over these local error clusters and reaches the correct path with fewer probes. In the architecture example, where the top-ranked path is already correct, early paths ($k = 1$–3) also yield correct labels (with $k = 3$ providing a particularly explicit explanation); even though some later correct paths are skipped, the correct label remains dominant in the aggregated confidence.

How early path backtracking works. Figure 4 illustrates how early backtracking prevents errors from becoming entrenched. In Path 1 (orange), the model drifts toward the incorrect span *"defensive end"*, with several local minima falling below the confidence threshold (yellow stars). Our rule treats these minima as warning signals and branches before the first one (at step 2), creating Path 2 (red), which instead converges to the correct answer *"linebacker"*. Once the erroneous span has been fully generated and reinforced by high-confidence tokens, later branching rarely fixes it, underscoring the importance of backtracking early.

5 Conclusion

We propose GCoT-Decoding, a general decoding strategy that extends earlier chain-of-thought based methods to broader QA tasks. By refining the branching mechanism for generating candidate



(a) Impact of decoding path count k.

(b) Effect of model size.

(c) Time cost and marginal gain rate by decoding path count k.

Figure 3: The impact of model size and the number of decoding paths k.

Table 7: Decoding outputs with confidence gaps $\Delta_{k,\text{answer}}$ for two classification examples.

	Question: AUSTRO-ITALIAN WAR, JACOBITE REBELLION, and FRANCO-SPANISH WAR are instances of		**Question:** Profitable home Chelisheva, The House with Lions, and House under the steeple can be classified as	
	Ground truth: historical wars		**Ground truth:** tourist attractions / architecture in Russia	
k=1	*European diplomatic initiatives.* So the answer is: **European diplomatic initiatives** (Δ=0.22)	×	*These are notable tourist attractions located across Russia.* So the answer is: **tourist attractions** (Δ=0.81)	✓
k=2	*diplomatic initiatives.* So the answer is: **diplomatic initiatives.** (Δ=0.18)	×	*architectural heritage in Russia.* So the answer is: **architecture in Russia** (Δ=0.68)	✓
k=3	*These events can be categorized under diplomatic initiatives.* So the answer is: **diplomatic initiatives** (Δ=0.09)	×	*tourist attractions in Russia. Explanation: each of these locations is a notable architectural site known for its historical significance within Russian cities.* So the answer is: **tourist attractions** (Δ=0.93)	✓
k=5	*They are relevant to international treaty formation.* So the answer is: **international treaty formation** (Δ=0.14)	×	*They refer to government-owned residential complexes.* So the answer is: **government-owned residential complexes** (Δ=0.24)	×
k=8	*Historical wars, because each conflict exemplifies armed struggles ...* So the answer is: **historical wars** (Δ=0.81)	✓	*metaphors from Soviet-era literature about class struggle.* So the answer is: **Soviet-era literature** (Δ=0.11)	×

Aggregation Module	Base Gen. Time	Overhead	Overhead %	MATCH
MaxPath (No agg.)	45.0 s	0.0 s	0.0%	21.8
Greedy Clustering (Ours)	45.0 s	0.2 s	$\sim$ 0.4%	54.6
LLM-based Aggregation	45.0 s	8.3 s	$\sim$ 18.4%	35.8

Table 8: Time cost breakdown per question (Gemma-7B, $k = 10$ paths) on GSM8K.

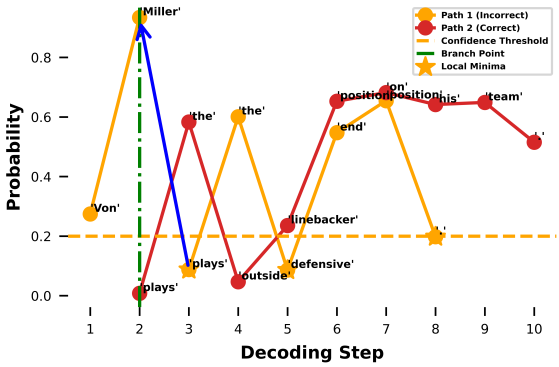


Figure 4: Illustration of early path backtracking.

paths, our approach further boosts performance. Experiments show that GCoT-Decoding consistently improves the reasoning ability of language models of various sizes and offers greater robustness to task drift across diverse benchmarks.

Limitations

Despite these benefits, GCoT-Decoding introduces additional computational overhead due to exploring and maintaining multiple reasoning paths. Our current evaluation is also limited to a set of QA and reasoning benchmarks, and does not fully cover tasks where reasoning is more implicit (e.g., summarization-style generation). Going forward, we are exploring optimizations such as early path pruning and more adaptive branching to reduce computational cost, as well as extending evaluation to a wider range of tasks that require step-by-step reasoning (e.g., structured text generation, logical inference, and multi-hop reasoning). For summarization-like tasks, we plan to investigate hybrid approaches that selectively apply GCoT-Decoding only to reasoning-intensive components, aiming to balance efficiency with broader applicability.

Acknowledgements

We appreciate the valuable discussion from the anonymous reviewers. This work was supported by the Solfeggio Ear-Training Intelligent Robot and Cloud Platform RD Project for Music Education (No. 2024CXY0102), the 3D

Visualization Digital Twin Integrated Control System Project(No. 2023CXY0111), the Pre-research Project for Introduced Talents of Minjiang University (No. MJY25025), the Public Technology Service Platform Project of Xiamen City(No. 3502Z20231043), and the Fujian Provincial Science and Technology Major Project (No. 2024HZ022003).

References

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. 2022. Ask me anything: A simple strategy for prompting language models. *arXiv preprint arXiv:2210.02441*.

Fahao Chen, Peng Li, Tom H Luan, Zhou Su, and Jing Deng. 2025. Spin: Accelerating large language model inference with heterogeneous speculative models. *arXiv preprint arXiv:2503.15921*.

Wenqing Chen, Weicheng Wang, Zhixuan Chu, Kui Ren, Zibin Zheng, and Zhichao Lu. 2024. Self-para-consistency: Improving reasoning tasks at low cost for large language models. In *62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, pages 14162–14167. Association for Computational Linguistics (ACL).

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Xueren Ge, Sahil Murtaza, Anthony Cortez, and Homa Alemzadeh. 2025. Expert-guided prompting and retrieval-augmented generation for emergency medical service question answering. *Preprint*, arXiv:2511.10900.

Xueren Ge, Sahil Murtaza, Anthony Cortez, and Homa Alemzadeh. 2026. Emsdialog: Synthetic multi-person emergency medical service dialogue generation from electronic patient care reports via multi-llm agents. *Preprint*, arXiv:2604.07549.

Olga Golovneva, Sean O'Brien, Ramakanth Pasunuru, Tianlu Wang, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Pathfinder: Guided search over multi-step reasoning paths. *arXiv preprint arXiv:2312.05180*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Hongru Ji, Yuyin Fan, Meng Zhao, Xianghua Li, Lianwei Wu, and Chao Gao. 2026. Stride-ed: A strategy-grounded stepwise reasoning framework for empathetic dialogue systems. *Preprint*, arXiv:2604.07100.

Fengqing Jiang. 2024. Identifying and mitigating vulnerabilities in llm-integrated applications. Master's thesis, University of Washington.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 36:36407–36433.

Hongye Liu and Ricardo Henao. 2025. Learning to substitute words with model-based score ranking. 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 11551–11565.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *International Conference on Machine Learning*, pages 30706–30775. PMLR.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Yuxi Sun, Aoqi Zuo, Haotian Xie, Wei Gao, Mingming Gong, and Jing Ma. 2026. Fact-e: Causality-inspired evaluation for trustworthy chain-of-thought reasoning. *Preprint*, arXiv:2604.10693.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025. Confidence improves self-consistency in llms. *arXiv preprint arXiv:2502.06233*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024. Soft self-consistency improves language model agents. *arXiv preprint arXiv:2402.13212*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022a. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650.

Jiaming Xu, Jiayi Pan, Yongkang Zhou, Siming Chen, Jinhao Li, Yaoxiu Lian, Junyi Wu, and Guohao Dai. 2025. Specee: Accelerating large language model inference with speculative early exiting. *arXiv preprint arXiv:2504.08850*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Liang Yao. 2024. Large language models are contrastive reasoners. *arXiv preprint arXiv:2403.08211*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H Chi, and Denny Zhou. 2023. Large language models as analogical reasoners. *arXiv preprint arXiv:2310.01714*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 35:30378–30392.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024a. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Yufeng Zhang, Xuepeng Wang, Lingxiang Wu, and Jinqiao Wang. 2024b. Enhancing chain of thought prompting in large language models via reasoning patterns. *arXiv preprint arXiv:2404.14812*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. 2024. Kg-cot: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 6642–6650. International Joint Conferences on Artificial Intelligence.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022a. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. 2024. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems*, 37:123846–123910.

A Additional Analysis of CoT-decoding

A.1 Sensitivity of CoT-decoding to answer-span extraction

A natural way to extend CoT-decoding (Wang and Zhou, 2024) beyond fixed-answer math questions is to ask the model to restate the final answer, e.g., by appending a marker such as *"So the answer is:"* and computing confidence on the continuation. However, when the same answer phrase appears multiple times in the trace, or when the wording of the marker varies slightly, different choices of "answer span" can lead to noticeably different confidence scores.

Table 9: Distribution of correct and incorrect paths and their corresponding confidences for the top 100 GSM8K questions in the case of first-index error.

Index	Correct	Incorrect	C. Conf.	I. Conf.
0	–	–	–	–
1	8	92	0.73	0.09
2	2	98	0.68	0.13
3	13	87	0.70	0.10
4	23	77	0.74	0.14
5	18	82	0.71	0.15
6	28	72	0.69	0.16
7	35	65	0.67	0.17
8	68	32	0.64	0.18
9	44	56	0.62	0.20

We compare two extraction methods on GSM8K, MultiArith, and the BBH *Sports Understanding* benchmark. The **rule-based** extractor follows the official evaluation protocols on these fixed-answer tasks: for GSM8K and MultiArith it takes the last number in the response, and for *Sports Understanding* it takes the final binary token (yes/no). The **prompt-based** extractor instead extends the model output with *"So the answer is:"* and uses the continuation as the answer span. As shown in Figure 5, simply switching from the rule-based span to the prompt-based span can substantially degrade CoT-decoding: on GSM8K and MultiArith it often collapses toward the greedy baseline, and on *Sports Understanding* it yields 5–12 point drops.

These results highlight two issues. First, CoT-decoding is structurally tied to a single, task-specific answer span, which limits its applicability to free-form QA where no canonical span exists. Second, even when such a span is available, small changes to how it is identified can have a surprisingly large impact. This motivates the confidence layer of GCoT-decoding (Section 3.2), where we treat each path as a "reasoning trace + answer continuation" rather than relying on a hand-picked span.

A.2 Greedy exploration can miss deeper correct paths

We also examine how CoT-decoding explores candidate paths. In the original formulation, paths are generated by perturbing only the first decoding step and then greedily rolling out each seed. Candidate paths are ordered by likelihood, and naive multi-path strategies simply take the first K of them. This can be a poor search strategy when early high-probability seeds form clusters of near-duplicate yet incorrect continuations.

Following Wang and Zhou (2024), we analyze the top 100 GSM8K questions under a controlled

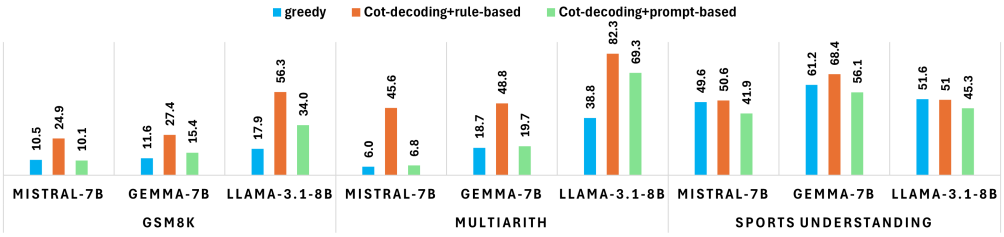


Figure 5: Impact of different answer extraction strategies on CoT-decoding performance.

setting where the first-ranked path is incorrect. For each index $i \in \{1, \ldots, 9\}$, we measure (i) how often the path at index i is correct vs. incorrect, and (ii) the average confidence of correct and incorrect paths at that index. Table 9 summarizes the distribution of correctness and confidence across indices. When the first index is wrong, the next few indices are also dominated by incorrect paths with similar surface forms and intermediate reasoning steps. Correct paths only become common at larger indices, so sweeping many adjacent early indices yields diminishing returns while consuming a large decoding budget.

This observation motivates the exploration layer of GCoT-decoding (Section 3.1): rather than exhaustively sweeping consecutive indices, we use Fibonacci indices to spread a fixed seed budget roughly log-uniformly over the ranked candidates, and add a local-minimum backtracking mechanism that spawns a few additional branches when the token-level confidence along a path exhibits a sharp local drop. Together, these mechanisms allow GCoT to reach deeper correct paths more efficiently than naive index-ordered exploration.

B Algorithm Details

We provide the pseudocode of path sampling and backtracking in Algorithm 1, and the pseudocode of the decoding path aggregation algorithm based on semantic clustering in Algorithm 2. We provide the pseudocode of path sampling and backtracking in Algorithm 1, and the pseudocode of the decoding path aggregation algorithm based on semantic clustering in Algorithm 2.

C Extended Ablation and Hyperparameter Sensitivity

C.1 Extended Ablation Study of GCoT-Decoding

As shown in Table 10, computing path confidence using the softmax probability gap between the top-

Algorithm 1: General decoding path generation with Fibonacci sampling and backtracking

Input: Model `model`, tokenizer `tokenizer`, query `query`, first branching size k, second branching size k', confidence threshold δ
Output: List of final decoding paths
Initialize empty result list $\mathcal{R}$;
// First branching Compute logits from initial `query` using `model`;
Select tokens at indices determined by Fibonacci sequence: $\{F_1, F_2, \ldots, F_k\}$;
foreach *token index* $i \in \{F_1, F_2, \ldots, F_k\}$ **do**
  Form initial decoding prefix by appending token t_i to `query`;
  Greedily decode from this prefix to obtain complete path $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ and token confidences $\{s_1, s_2, \ldots, s_T\}$;
  Append path and confidences to temporary list $\mathcal{L}$;
end
// Secondary branching via backtracking **foreach** *decoded path* $\mathbf{y}$ *and confidences* $\{s_t\}_{t=1}^{T}$ *in* $\mathcal{L}$ **do**
  Identify local minima set $S = \{t \mid 3 \le t \le T, s_t < s_{t-1}, (t < T \Rightarrow s_t < s_{t+1}), s_t < \delta\}$;
  Determine branching point b:

$$b = \begin{cases} \min S, & S \neq \varnothing \\ -1, & S = \varnothing \end{cases}$$

  if $b \neq -1$ **then**
    Truncate path to form prefix $\mathbf{y}_{<b} = (y_1, \ldots, y_{b-2})$;
    Compute logits for next token after prefix $\mathbf{y}_{<b}$;
    Select alternative tokens at Fibonacci indices $\{F_1, \ldots, F_{k'}\}$;
    foreach *alternative token index* $j \in \{F_1, \ldots, F_{k'}\}$ **do**
      Append token $y_{b-1}^{(j)}$ to prefix $\mathbf{y}_{<b}$;
      Greedily decode from new prefix to complete new path $\mathbf{y}^{(j)}$;
      Add new path $\mathbf{y}^{(j)}$ to result list $\mathcal{R}$;
    end
  end
  else
    Add original path $\mathbf{y}$ directly to result list $\mathcal{R}$;
  end
end
return result list $\mathcal{R}$

Algorithm 2: General decoding path aggregation via semantic clustering

Input: Decoding paths $\{p_i\}_{i=1}^{K}$, confidences $\{c_i\}_{i=1}^{K}$, embedding function $\phi(\cdot)$, similarity threshold τ
Output: Final aggregated answer
Initialize semantic groups: $G_j \leftarrow \emptyset$, representatives $r_j \leftarrow \emptyset$, group count $N \leftarrow 0$;
foreach *path output* $g_i = \text{gen}_2(p_i)$ **do**
  Compute embedding $\phi(g_i)$;
  if *N = 0* **then**
    Create new group $G_1 = \{g_i\}$, set representative $r_1 = g_i$, set $N = 1$;
    continue;
  end
  Compute similarities $s_{i,j} = \cos(\phi(g_i), \phi(r_j))$ for all existing groups $j = 1, \ldots, N$;
  Find the minimal index j^* satisfying $s_{i,j^*} \geq \tau$; if none exist, set $j^* = N + 1$;
  if $j^* \leq N$ **then**
    Add g_i to existing group G_{j^*};
  else
    Create new group $G_{N+1} = \{g_i\}$, set representative $r_{N+1} = g_i$, increment N;
  end
end
Compute cumulative confidence $C_j = \sum_{g_i \in G_j} c_i$ for each group j;
Select group with maximum cumulative confidence $j_{\max} = \arg\max_j C_j$;
Return group representative $r_{j_{\max}}$ as the final output.

2 tokens consistently outperforms raw logits and entropy across tasks. While raw logits are sensitive to distributional shifts—especially in open-ended QA—entropy tends to misrepresent confidence due to token fragmentation in LLMs. For path generation, replacing Fibonacci sampling with sequential decoding reduces the likelihood of reaching correct answers, and removing the second-stage backtracking prevents correction of low-confidence tokens, allowing flawed reasoning paths to persist. Moreover, selecting only the highest-confidence path (MaxPath) significantly undermines decoding stability; on SQuAD, this leads to performance drops of up to 16.4%. These results underscore the importance of multi-path aggregation in mitigating single-path errors and capturing diverse yet valid reasoning chains, which are essential for robust GCoT-Decoding. Overall, when multiple components are simultaneously simplified, the performance of GCoT-Decoding deteriorates rapidly, underscoring the importance of all three modules working in concert.

C.2 Sensitivity to Hyperparameters

We also provide sensitivity experiments on the similarity threshold τ and the confidence threshold δ, summarized in Table 11.

D Prompt Demonstration Examples

Figure 6 shows the chain-of-thought prompting examples we use for the SQuAD `dev-v1.1` task. In the **zero-shot** setting, no demonstrations are provided. The **one-shot** setting includes only Example 1, while the **three-shot** setting incorporates all three examples.

Example 1
Context: The Hubble Space Telescope was launched into low Earth orbit in 1990 aboard the Space Shuttle Discovery. It has since captured landmark images such as the Hubble Ultra-Deep Field, revealing thousands of distant galaxies. In 2009, the final servicing mission upgraded its cameras and sensors.
Question: Which space telescope captured the Ultra-Deep Field image?
Answer: Hubble Space Telescope

Example 2
Context: 'The Lord of the Rings' is a high-fantasy trilogy originally published in three volumes between 1954 and 1955. Written by J.R.R. Tolkien, it follows the quest of Frodo Baggins to destroy the One Ring and defeat the Dark Lord Sauron.
Question: Who is the author of 'The Lord of the Rings'?
Answer: J.R.R. Tolkien

Example 3
Context: Ratatouille is a classic vegetable stew from southern France, typically including eggplant, zucchini, bell peppers, tomatoes, onions, and garlic, flavored with herbs de Provence. It is named after a city on the Côte d'Azur where it originated.
Question: Which French city is ratatouille traditionally associated with?
Answer: Nice

Figure 6: Prompting examples used in the SQuAD `dev-v1.1` task under different few-shot settings. Zero-shot uses no demonstrations, one-shot includes only Example 1, and three-shot includes all three examples.

Figure 7 shows the chain-of-thought demonstrations used for the GSM8K task. Similarly, the **zero-shot** configuration contains no examples, the **one-shot** configuration includes only the first example, and the **three-shot** configuration includes all three. These prompts are used to evaluate the effect of demonstration count on arithmetic reasoning performance.

E Analysis on Clustering and Representative Selection

As shown in Table 12, while different clustering algorithms (Greedy, K-Means++, Agglomerative, Spectral) yield nearly identical accuracies, the representative selection strategy makes a substantial difference. Specifically, choosing the first-in-cluster answer consistently outperforms alternatives such as selecting the cluster centroid or the maximum-confidence path. This confirms that index ordering plays a crucial role in GCoT-decoding, and that a greedy clustering scheme combined with first-in-cluster selection is both efficient and effective.

Confidence Computation	Path Generation	Path Aggregation	GSM8K (Acc., Gemma-7B)	GSM8K (Acc., Mistral-7B)	SQuAD v1.1 (MATCH, Gemma-7B)	SQuAD v1.1 (MATCH, Llama-3.1-8B)
–	–	–	**21.8**	**18.0**	**54.6**	**67.2**
entropy	–	–	18.3 (–3.5)	14.1 (–3.9)	51.4 (–3.2)	63.0 (–4.2)
logits	–	–	19.0 (–2.8)	15.7 (–2.3)	54.5 (–0.1)	66.9 (–0.3)
–	Seq	–	17.5 (–4.3)	13.1 (–4.9)	48.9 (–5.7)	63.6 (–3.6)
–	OneBranch	–	20.8 (–1.0)	16.5 (–1.5)	52.7 (–1.9)	67.0 (–0.2)
–	–	MaxPath	15.3 (–6.5)	12.8 (–5.2)	41.9 (–12.7)	50.8 (–16.4)
entropy	Seq	–	13.9 (–7.9)	11.2 (–6.8)	42.1 (–12.5)	49.4 (–17.8)
logits	–	MaxPath	12.5 (–9.3)	10.7 (–7.3)	39.7 (–14.9)	45.6 (–21.6)
–	Seq	MaxPath	12.2 (–9.6)	9.9 (–8.1)	36.3 (–18.3)	44.7 (–22.5)
entropy	–	MaxPath	12.8 (–9.0)	10.3 (–7.7)	40.8 (–13.8)	46.5 (–20.7)
–	OneBranch	MaxPath	11.5 (–10.3)	8.8 (–9.2)	33.6 (–21.0)	41.9 (–25.3)
entropy	Seq	MaxPath	8.4 (–13.4)	6.7 (–11.3)	25.8 (–28.8)	33.0 (–34.2)
logits	Seq	MaxPath	8.9 (–12.9)	7.4 (–10.6)	27.5 (–27.1)	34.6 (–32.6)
entropy	OneBranch	MaxPath	7.7 (–14.1)	5.4 (–12.6)	19.5 (–35.1)	24.8 (–42.4)

Table 10: Ablation study of GCoT-Decoding. Top rows show single-factor ablations; bottom rows show selected multi-factor variants. Numbers in parentheses denote drops from the full model.

Table 11: Performance under different thresholds τ and δ on GSM8K, MultiArith, and Sports Understanding tasks.

τ	δ	GSM8K			MultiArith			Sports Underst.		
		Mistral-7b	Gemma-7b	Llama-3.1-8b	Mistral-7b	Gemma-7b	Llama-3.1-8b	Mistral-7b	Gemma-7b	Llama-3.1-8b
0.8	0.2	18.0	21.8	41.7	31.3	23.2	74.3	52.0	65.2	58.0
0.7	0.2	16.9	20.5	40.8	30.1	21.9	72.6	49.8	63.7	55.3
0.9	0.2	17.3	21.0	40.9	30.5	22.7	73.2	51.5	64.2	57.0
0.8	0.1	17.2	21.1	41.1	30.7	22.4	73.4	51.7	64.5	57.3
0.8	0.3	17.4	21.4	41.2	30.6	22.6	73.7	51.6	64.7	57.5

Example 1
Question: Tobias is buying a new pair of shoes that costs $95. He has been saving up his money each month for the past three months. He gets a $5 allowance a month. He also mows lawns and shovels driveways. He charges $15 to mow a lawn and $7 to shovel. After buying the shoes, he has $15 in change. If he mows 4 lawns, how many driveways did he shovel?
Answer:
He saved up $110 total because 95 + 15 = <<95+15=110>>110.
He saved $15 from his allowance because 3 × 5 = <<3*5=15>>15.
He earned $60 mowing lawns because 4 × 15 = <<4*15=60>>60.
He earned $35 shoveling driveways because 110 - 60 - 15 = <<110-60-15=35>>35.
He shoveled 5 driveways because 35 ÷ 7 = <<35/7=5>>5.
Final Answer: 5

Example 2
Question: Emma wants to buy a bicycle that costs $120. She has been saving her weekly allowance of $8 for the past 5 weeks. She also walks dogs and earns $12 per dog. After buying the bicycle, she has $20 left. If she walked 6 dogs, how many additional odd jobs did she do if she earns $5 per odd job?
Answer:
She saved up $140 total because 120 + 20 = <<120+20=140>>140.
She saved $40 from her allowance because 5 × 8 = <<5*8=40>>40.
She earned $72 walking dogs because 6 × 12 = <<6*12=72>>72.
She earned $28 from odd jobs because 140 - 72 - 40 = <<140-72-40=28>>28.
She did 5 odd jobs because 28 ÷ 5 = <<28/5=5.6>>5.6 (rounded to 5).
Final Answer: 5

Example 3
Question: Liam is purchasing a video game console for $180. He saved his monthly allowance of $10 for 4 months. He also tutors kids for $15 per session. After the purchase, he has $30 remaining. If he tutored 8 times, how many times did he babysit if he earns $12 per babysitting job?
Answer:
He saved up $210 total because 180 + 30 = <<180+30=210>>210.
He saved $40 from his allowance because 4 × 10 = <<4*10=40>>40.
He earned $120 tutoring because 8 × 15 = <<8*15=120>>120.
He earned $50 babysitting because 210 - 120 - 40 = <<210-120-40=50>>50.
He babysat 4 times because 50 ÷ 12 = <<50/12=4.166>>4 (rounded down).
Final Answer: 4

Figure 7: Prompting examples used in different few-shot settings for the GSM8K task, adapted to arithmetic reasoning.

F Choice of the number of backtracking

We find that CoT errors tend to have early turning points: as soon as the model commits to a wrong semantic decision (Table 16), the token-level confidence exhibits a sharp local drop, and subsequent tokens mostly elaborate on this misconception rather than correcting it. In these cases, back-

Table 12: Accuracy comparison of clustering methods and representative choices on SQuAD v1.1.

Category	Method	Gemma-7B	Llama-3.1-8B
Clustering	Greedy Clustering	54.6	**67.2**
	K-Means++	54.4	66.9
	Agglomerative (Ward)	**54.7**	67.1
	Spectral Clustering	54.5	67.0
Representative	First-in-Cluster	**54.6**	**67.2**
	Cluster Centroid	47.8	60.4
	Max-Conf	48.2	60.9

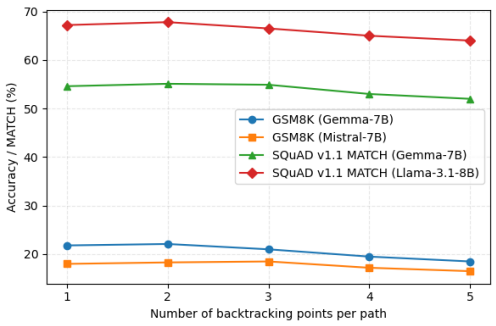


Figure 8: Effect of the maximum number of backtracking points per path under a fixed overall path budget.

Method	GSM8K (Acc.)			MultiArith (Acc.)			Sports Understanding (Acc.)		
	Mistral-7B	Gemma-7B	Llama-3.1-8B	Mistral-7B	Gemma-7B	Llama-3.1-8B	Mistral-7B	Gemma-7B	Llama-3.1-8B
GCoT-decoding + SpanAlign (Last)	10.7	15.4	34.0	16.8	19.7	69.3	48.0	67.2	52.0
GCoT-decoding + SpanAlign (Mean)	10.2	14.9	33.5	16.1	19.0	68.5	47.1	66.3	51.4

Table 13: Comparison between using only the last aligned answer span (SpanAlign (Last)) and averaging over all aligned spans (SpanAlign (Mean)).

Template	SQuAD v1.1 MATCH (Gemma-7B)
"So the answer is . . . "	54.6
"Therefore, the answer is . . . "	54.5
"Final answer:"	54.3

Table 14: Ablation on answer-extraction templates for GCoT-decoding on SQuAD v1.1.

Setting	SQuAD v1.1 BLEU	SQuAD v1.1 MATCH	Auto-cat BLEU	Auto-cat MATCH
GCoT + MiniLM	10.0	67.2	10.6	30.5
GCoT + MPNet-base	9.8	66.7	10.4	30.3
GCoT + E5-small	10.1	67.0	10.5	30.4

Table 15: Embedding model ablation for the semantic clustering module in GCoT-decoding.

tracking at the first confidence valley is typically sufficient to redirect the reasoning towards a different, potentially correct branch. From an efficiency perspective, allowing multiple backtracking points per path under a fixed path budget significantly increases decoding cost and complicates how to trade off early vs. late corrections, so we adopt a simple one-shot backtracking rule as a pragmatic accuracy–efficiency compromise.

Figure 8 summarizes this ablation by varying the maximum number of backtracking points per path from 1 to 5: performance improves slightly from 1-back to 2-back, stays roughly flat around 3-back, and then drops noticeably at 4 and 5. This pattern indicates that limited extra backtracking offers only marginal gains, while aggressive multi-backtracking quickly hurts both accuracy and efficiency, supporting our choice of a single-shot local-minima strategy.

G Effect of answer-extraction templates

In Section 3.2, we use a short continuation template (e.g., "So the answer is ...") purely as an answer-extraction marker after the model has already produced a full chain-of-thought reasoning trace. To verify that GCoT-decoding does not depend on the specific wording of this marker, we evaluate several semantically equivalent templates on SQuAD v1.1 with Gemma-7B, while keeping all other components fixed (Table 14).

The variation across templates is within 0.3 absolute MATCH points, which is negligible compared to the gains obtained by switching from greedy or vanilla CoT-decoding to GCoT on the same benchmark. This supports our claim that GCoT-decoding does not hinge on a specific wording of the answer-extraction template.

H Embedding model ablation for semantic clustering

GCoT-decoding uses an off-the-shelf sentence embedding model to perform greedy semantic clustering over candidate paths. To assess the sensitivity of this module to the choice of embedding space, we fix the rest of the framework and only vary the embedding model, comparing MiniLM, MPNet-base, and E5-small on SQuAD v1.1 and Auto-Categorization (Table 15).

Across all settings, the variation in BLEU and MATCH is within 0.5 absolute points, suggesting that the greedy clustering module is relatively insensitive to the specific off-the-shelf embedding model used, as long as it provides a reasonable semantic similarity signal. This matches our design goal of treating semantic clustering as a conservative, pluggable enhancement over simple max-path selection.

I SpanAlign ablation: last vs. mean alignment

In Section 3.2, we use an LCS-based SPANALIGN module to compare answer segments across different paths. When the same answer phrase appears multiple times in a reasoning trace, our default implementation scores only the terminal aligned segment ("SpanAlign (Last)"). To check whether averaging over all aligned segments could be preferable, we compare this default against a variant that averages confidence across all occurrences ("SpanAlign (Mean)") on GSM8K, MultiArith, and Sports Understanding (Table 13).

Across all three datasets and models, using the final occurrence of the aligned answer span is at least as reliable as averaging over all occurrences,

Table 16: An example of path backtracking. The underlined segments indicate the answers targeted by the decoding paths, while the highlighted portions show the content generated after backtracking. “plays”, “defensive”, and “.” are the three local minima in Path1.

Question: *What position does Von Miller play?*
Path1(×): *Von Miller plays(0.0877)* ***defensive(0.0921) end*** *position .(0.1980)*
Path2(✓): *Von plays the* ***outside linebacker*** *position on his team .*
Path3(×): *Von Miller plays the* ***defensive end*** *role for his team and is known for his pass rushing ability .*

and often slightly better.

J Qualitative example of early path backtracking

We provide a qualitative example in Table 16 to illustrate early error correction in the decoding process. In Path1, the incorrect answer “defensive end” emerges after three local minima. Branching before the first error token (e.g., at “plays”) allows effective correction, as in Path2, which leads to the correct answer “linebacker.” In contrast, branching after the error fragment has formed, as in Path3, fails to revise the mistake—once embedded, the error resists recovery. This highlights the importance of early branching before erroneous spans are committed.