DISCOVERING DEEP CHAIN-OF-THOUGHT PATHS ACROSS BROADER QA: A GENERAL COT-DECODING FRAMEWORK FOR LLMS

Anonymous authors
Paper under double-blind review

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 and effectiveness.

1 INTRODUCTION

Introducing Chain-of-Thought (CoT) can effectively enhance the reasoning capability of large language models (LLMs). Existing studies primarily guide models to generate CoT paths through prompt engineering (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). However, prompts can be influenced by the biases of their designers, and distinct tasks require different prompt designs (Wang et al., 2022b; Ye & Durrett, 2022; Zhou et al., 2022b), thus limiting their generality. Recent research has also aimed to enhance the reasoning capabilities of language models from a decoding perspective, such as self-consistency methods (Wang et al., 2022a), contrastive decoding (Li et al., 2022), and context-aware decoding (Shi et al., 2024). Nevertheless, these methods usually require additional information.

Therefore, the question arises: Can large language models perform Chain-of-Thought reasoning without prompts? Wang & Zhou (2024) propose a prompt-free CoT-decoding approach, which explores the top-k alternative tokens for the first token in the decoding path, identifies specific answer spans from these paths, computes the difference between their logits as the confidence, and aggregates the paths pointing to the same answer. The answer with the highest cumulative confidence is selected as the final result.

However, CoT-decoding heavily relies on specific answer spans to accurately compute path confidence and to aggregate paths pointing to the same answer. As shown in Table 1, in the fixed-format GSM8K-style toy-production problem on the right, all top-k CoT paths end with the same numeric span "24", so CoT-decoding can simply match this exact span and aggregate paths that point to it. In contrast, in the free-form question about U.S. presidents on the left, the same correct answer is phrased differently , while another path mentions a plausible but wrong alternative, so there is no single canonical span for exact matching or majority voting. Moreover, CoT-decoding performs branching solely at the first token in index order, which makes it difficult to discover correct paths that lie deeper in the decoding sequence, while early decoding errors can further disrupt the generation of correct paths.

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 were U.S. presidents. **k=2**: These men were Civil War generals. **k=3**: All served as presidents.	**Q**: A factory makes 3 toys per hour. How many toys after 8 hours? **k=1**: $3 \times 8 = \underline{24}$ (0.93) **k=2**: 3 times 8 is $\underline{24}$ (0.91) **k=3**: = $\underline{24}$ (0.85)
Answer Space	∞	$\mathbb{N}$
Exact Span Match	$\times$	✓
Majority Vote Aggregation	$\times$	✓

To address this issue, we propose General Chain-of-Thought Decoding **(GCoT-Decoding)** that can effectively identify decoding paths containing CoT reasoning without relying on specific answer spans, thereby extending applicability to a broader range of question-answering tasks. Specifically, we introduce a novel two-stage branching strategy: in the first stage, we perform Fibonacci sampling at an early decoding step to select k alternative tokens, ensuring path diversity; in the second stage, we backtrack to the token with locally minimal confidence to correct potentially erroneous paths, followed by greedy decoding to complete the remaining steps.

For confidence computation, we insert an additional prompt after the model's initial output to explicitly extract the final answer. It is important to note that this prompt is fundamentally different from conventional CoT prompts—it serves solely to extract the final answer from the model's response, without guiding or influencing the reasoning process, and thus does not affect the final reasoning outcome. We use the length-normalized top-2 logits gap as the confidence score for the final answer, as CoT paths typically involve longer reasoning steps. Finally, we aggregate similar paths based on semantic similarity and select the earliest path within the group with the highest cumulative confidence; the answer indicated by this path is taken as the final output.

We conduct comprehensive experiments on six datasets across both fixed and free-form QA tasks. Our method significantly improves performance on free QA while preserving strong performance on fixed QA. Additionally, it can be combined with prompting to further enhance reasoning capabilities. Overall, our contributions are summarized as follows:

- **Proposing GCoT-Decoding**: A novel and general decoding strategy that does not rely on specific answer spans, thereby improving adaptability to diverse question-answering tasks.
- **Optimizing the branching strategy**: By introducing a two-stage branching mechanism, our method more efficiently discovers correct answers hidden in later decoding steps while correcting potentially erroneous paths.
- **Efficient path aggregation method**: We adopt a semantic similarity–based clustering strategy with a fixed threshold, and select the earliest path in each cluster as the representative. Compared to using the cluster centroid or the most similar path, this design simplifies computation while maintaining performance.

2 Motivation

2.1 CoT-decoding relies heavily on specific answer spans

To extend CoT-decoding to free-form QA tasks, a natural idea is to explicitly guide the model to output the final answer by including prompts such as *"So the answer is:"*, thereby replacing the rule-matched answer spans typically used in fixed-format QA. However, when the same answer appears multiple times along the decoding path, selecting different answer spans can still lead to inconsistencies in confidence calculation.

To quantitatively investigate the effect of specific answer spans on CoT-decoding, we apply two different methods for extracting answers across GSM8K, MultiArith, and the BBH *Sports Understanding* benchmark: (1) a **rule-based method**, aligned with the official evaluation protocols on these fixed-answer tasks, which for numeric math benchmarks such as GSM8K and MultiArith identifies

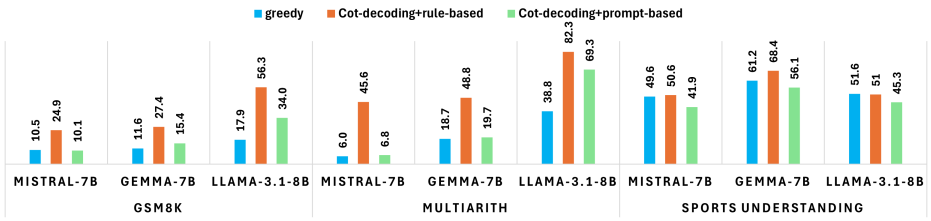


Figure 1: Impact of different answer extraction strategies on CoT-decoding performance.

the last number in the response and computes the average token confidence for that span, while for *Sports Understanding* it identifies the final binary answer token ("yes" or "no") in the model's continuation; (2) a **prompt-based method**, which uses the template *"So the answer is:"* to extend the model's output and compute the average confidence of the answer tokens in the extended segment. As shown in Figure 1, using the prompt-based method to identify answer spans significantly reduces the performance of CoT-decoding: on GSM8K and MultiArith, it often collapses back toward the greedy baseline, and on *Sports Understanding* it yields 5–12 point drops compared with the rule-based extractor. In contrast, while the rule-based method performs well, it relies on task-specific heuristics and thus lacks generality and cannot be applied to tasks with a broader or more open-ended answer space. Therefore, it is necessary to improve CoT-decoding strategies to better adapt to prompt-based approaches.

2.2 CoT-decoding can miss correct answers hidden in deeper paths

The reasoning ability of LLMs can be obscured by greedy decoding, which tends to yield a direct answer. By substituting the first decoding-step token with a lower-probability token, one may uncover the correct chain-of-thought path (Wang & Zhou, 2024).

Table 2: Distribution of correct and incorrect paths and their corresponding confidences for the top 100 questions in the GSM8K dataset 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	0.66	0.17
6	28	72	0.60	0.11
7	35	65	0.78	0.01
8	68	32	0.67	0.18
9	44	56	0.62	0.20

In CoT-decoding, candidate paths are ranked by their likelihoods, and naive strategies explore them sequentially from index $k{=}0$ upward. However, this strategy can be suboptimal, especially when the most probable early paths converge on the same incorrect answer. As shown in Table 2, when the first index is incorrect, the subsequent early-ranked paths tend to replicate the same error pattern. This occurs due to the high probability mass being distributed across semantically similar but incorrect continuations. The correct answer, in such cases, is often buried deeper among lower-probability candidates—hidden in higher-indexed paths, and increasing the number of explored paths would require a substantial amount of additional decoding time. Thus, an effective sampling strategy should avoid redundant exploration of adjacent early paths and instead prioritize diversity across the decoding space.

3 Method

In this section, we first present a two-stage branching strategy for generating candidate paths **(Sec. 3.1)**, then introduce a scoring scheme that combines path length with the top-2 logit gap to assign confidence to each candidate **(Sec. 3.2)**, and finally show how to aggregate free-form answers to mitigate the impact of small differences in model logits **(Sec. 3.3)**.

3.1 Two-stage path branching strategy

Fibonacci sampling of alternative tokens. We propose a Fibonacci sampling strategy, in which we first sort all candidate tokens at the first decoding step based on their model-assigned confidence scores. Then, instead of selecting the top-k tokens sequentially, we use indices from the Fibonacci

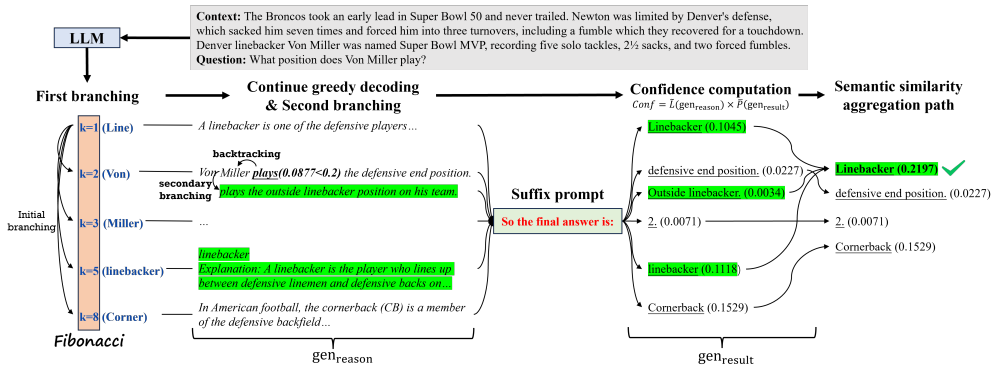


Figure 2: Overall Workflow of GCoT-Decoding. It generates candidate decoding paths via a two-stage branching strategy and then aggregates these paths based on semantic similarity.

sequence to choose k alternative tokens as initial branching points:

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

For each selected token, the model continues decoding the rest of the sequence using greedy decoding, thereby constructing a diverse set of candidate paths.

As shown in Table 2, when the first index is incorrect, Fibonacci sampling helps skip over these local error clusters and increases the chances of exploring correct paths in the tail. Even when the first index is correct, Fibonacci sampling may include some incorrect paths; however, these paths typically have lower confidence due to the lack of a clear reasoning chain. In contrast, the correct path, even if ranked lower, usually has a higher cumulative confidence and is more likely to be selected during the aggregation stage.

Backtracking from local minima. We observe that when the model starts to drift toward an incorrect answer during decoding, it often shows significantly lower confidence in certain tokens. Based on this, we propose a simple yet effective secondary branching strategy: identify the first point where the model's confidence drops noticeably, then backtrack and regenerate new paths from that point. Specifically, for a greedy decoding path $\mathbf{y} = (y_1, y_2, \ldots, y_T)$, we monitor the token-level confidence $s_t = P(y_t \mid x, y_{<t})$. We scan the path from step 3 onward to find the first local minimum where the confidence drops below a threshold δ. Formally, collect candidates:

$$S = \left\{ t \;\middle|\; 3 \le t \le T,\ s_t < s_{t-1},\ (t < T \Rightarrow s_t < s_{t+1}),\ s_t < \delta \right\}, \tag{2}$$

and define the backtracking index:

$$b = \begin{cases} \min S, & S \neq \varnothing, \\ -1, & S = \varnothing. \end{cases} \tag{3}$$

If $b \neq -1$, step back to y_{b-1} and branch on k' alternatives (e.g., Fibonacci indices) to form prefixes:

$$\mathbf{y}_{<b}^{(m)} = (y_1, \ldots, y_{b-2}, y_{b-1}^{(m)}), \qquad m \in \{F_1, \ldots, F_{k'}\}, \tag{4}$$

then complete each with greedy decoding, yielding the new candidate set:

$$\mathcal{P} = \{\mathbf{y}^{(m)}\}_{m=1}^{k'}. \tag{5}$$

This strategy allows us to explore meaningful alternatives near early signs of error while keeping the search efficient. Please see Appendix A for the pseudocode corresponding to this section.

3.2 Length-Aware Logit Gap Confidence

CoT-decoding calculates the average difference between the top-1 and top-2 softmax logits for each token in the answer span, treating it as the confidence score of the decoding path. This is based on the observation that the presence of a CoT path typically leads to more confident decoding of the final

answer, characterized by a significant probability gap Δ between the top and secondary predictions (Wang & Zhou, 2024). This method relies on answer tokens extracted by rules and is unsuitable for questions whose answer sets or formats are not fixed.

To address the above issue, we extend the original response gen_1 by appending the prompt *"So the answer is:"* to generate the final answer gen_2. This short template is used purely as a post-hoc answer extractor after the model has already produced the full reasoning trace, and does not guide the structure of the reasoning itself; in Appendix F we further show that replacing it with semantically equivalent phrases leads to only minor variations. Since relying solely on gen_2 may lead to large confidence deviations, we treat gen_1 as the reasoning part and gen_2 as the result part, and compute the confidence by multiplying the normalized length of gen_1 with the average logits gap Δ between the top-2 tokens in gen_2, since CoT paths typically involve longer reasoning steps. The new confidence calculation is formalized as follows:

$$\mathrm{GCoT}_{\Delta(k,\text{answer})} = \frac{\log\big(1+|\mathrm{gen}_{1_k}|\big)}{\max_{i\in K}\ \log\big(|\mathrm{gen}_{1_i}|\big)} \times \frac{1}{|\mathrm{gen}_2|}\sum_{x_t\in \mathrm{gen}_2}\big[p(x_t^1)-p(x_t^2)\big]\,. \tag{6}$$

Here, x_t^1 and x_t^2 represent the top-2 tokens at the t-th decoding step in the k-th decoding path. We also design an alignment method that finds the longest common subsequence $\mathrm{LCS}(\mathrm{gen}_1,\ \mathrm{gen}_2) = (s_{11}, s_{12}, \ldots, s_{1m}; s_{21}, s_{22}, \ldots, s_{2n})$, whose length is L. Before computing this LCS alignment, we normalize both generations by lowercasing and stripping pure punctuation tokens, which greatly reduces sensitivity to minor tokenization or punctuation differences. The final LCS in gen_1 is s_{1m}, the final LCS in gen_2 is s_{2n}, and we sum their average confidences:

$$\mathrm{GCoT+SpanAlign}_{\Delta(k,\text{answer})} = \frac{1}{L}\left(\sum_{x_{1t}\in s_{1m}}\Big[p\big(x_{1t}^1\big)-p\big(x_{1t}^2\big)\Big] + \sum_{x_{2t}\in s_{2n}}\Big[p\big(x_{2t}^1\big)-p\big(x_{2t}^2\big)\Big]\right). \tag{7}$$

When the same answer phrase is mentioned multiple times in a trace, we compare this default last-span scoring rule against a variant that averages over all aligned spans and observe very similar performance; detailed numbers are given in Appendix H.

3.3 Greedy Semantic Clustering for Path Aggregation

When relying solely on the path with the maximum Δ, small differences in the model's logits can have a significant impact on the results, whereas aggregation can mitigate this sensitivity (Wang & Zhou, 2024). However, for questions without a fixed answer set or output format, majority voting based on exact string matching is infeasible. Introducing semantic similarity for clustering brings a new challenge: semantically similar paths may still point to different answers.

We find that the most critical factor affecting aggregation is the choice of representative answer, rather than the clustering method itself. Although common practices include selecting the cluster centroid or the path with the highest confidence, GCoT-decoding is highly sensitive to index order, and selecting the earliest-indexed path yields better results. Thus, we adopt a greedy clustering method based on index ordering to aggregate the decoding paths, ensuring both the efficiency and effectiveness of the aggregation process. We provide the impact of different clustering methods and answer selection strategies in Appendix D. We further ablate the underlying sentence embedding model and find that GCoT's semantic clustering is largely insensitive to the specific encoder used (Appendix G).

Specifically, we denote all decoding paths as $\{p_i\}_{i=1}^K$, with each path producing a final answer $g_i = \mathrm{gen}_2(p_i)$ and associated confidence score $c_i = \mathrm{confidence}(p_i)$. We maintain a set of semantic groups $\{G_j\}_{j=1}^N$ with corresponding representative answers $\{r_j\}_{j=1}^N$, initialized as empty. For each answer g_i, we compute its cosine similarity with all existing representatives:

$$s_{i,j} = \cos\big(\phi(g_i),\ \phi(r_j)\big), \quad j = 1, 2, \ldots, N, \tag{8}$$

where $\phi(\cdot)$ is the embedding function, and the greedy assignment rule is defined as:

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

which always assigns g_i to the first eligible group according to index ordering. Then 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 assigned, we compute cumulative confidence for each group:

$$C_j = \sum_{g_i \in G_j} c_i, \quad j = 1, 2, \ldots, N, \tag{11}$$

and select the representative of the group with the highest cumulative confidence $r_{j_{\max}}, j_{\max} = \arg\max_j C_j$ as the final output. The pseudocode is given in Appendix A.

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 (e.g., integers or yes/no), including **GSM8K** and **MultiArith** (Cobbe et al., 2021; Roy & 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 & 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 & 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. For the sensitivity experiments on hyperparameter settings, please refer to Appendix B.

4.2 Main results

Fixed QA. As shown in Table 3, GCoT-decoding outperforms all single-path decoding strategies (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, as discussed in Section 2.1. 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 4, GCoT-decoding achieves the highest BLEU and MATCH scores in nearly all settings, significantly outperforming other methods in both generation quality and

	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 3: 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 4: 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.

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.

Free QA. As shown in Table 4, 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.

Results on a reasoning-tuned model. On GSM8K, we also evaluate the reasoning-tuned DeepSeek-R1 with its recommended math prompt. Even on this near-saturated benchmark (temperature sampling already reaches 96.1%), GCoT-decoding still improves over self-consistency and CoT-decoding, raising accuracy from 97.2% to 99.0%, which shows that it brings gains on top of strong reasoning-tuned models rather than exploiting weaknesses of smaller ones.

Decoding strategy	**GSM8K Acc. (DeepSeek-R1)**
Temperature sampling (T=0.7)	96.1
Self-consistency (k=10)	96.4
CoT-decoding (K=10)	97.2
GCoT-decoding (K=10)	**99.0**

Table 5: Accuracy of different decoding strategies on GSM8K with the reasoning-tuned model DeepSeek-R1.

4.3 Compatibility of GCoT-decoding with Prompting Methods

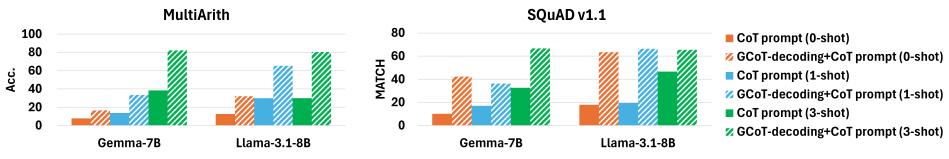


Figure 3: The results of combining GCoT-decoding with CoT prompting.

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 3) 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 C.

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. We also report additional ablations on confidence computation in Appendix B.

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 6 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.

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

Reliability of local-minima backtracking. We assess reliability by measuring trigger frequency and success rate on SQuAD v1.1 (Table 7). 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%), indicating that naive perturbations are not helpful. We further study the effect of allowing more backtracking points per path in Appendix E.

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

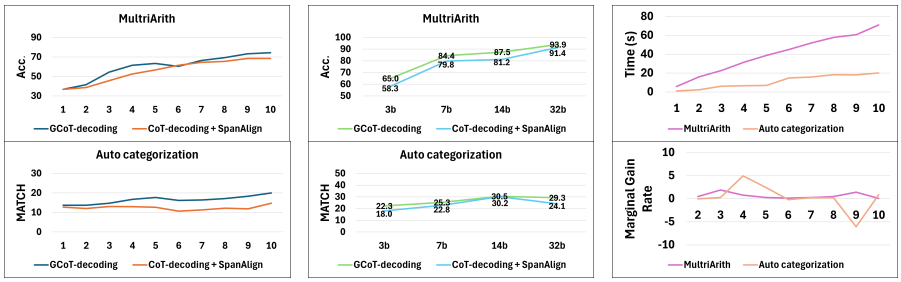
Greedy semantic clustering vs. LLM-based aggregation. We first compare GCoT-decoding with a MaxPath baseline that simply selects the single highest-confidence path: as shown in Table 8, greedy semantic clustering improves GSM8K accuracy from 15.3 to 21.8 and SQuAD MATCH from 41.9 to 54.6, with only 0.2 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.

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

4.5 Quantitative and Qualitative Analysis

Quantitative analysis. As shown in Figure 4(a), 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. Figure 4(b) shows that increasing the number of



(a) Effect of model size. (b) Impact of decoding path count k. (c) Time cost and marginal gain rate by decoding path count k.

Figure 4: The impact of model size and the number of decoding paths k.

decoding paths k initially improves performance but saturates after $k > 5$. *GCoT-decoding* maintains stronger and more stable gains than +*SpanAlign* across all k settings. As shown in Figure 4(c), time cost grows roughly linearly with k, while both tasks exhibit diminishing marginal gains. Taken together, the optimal "elbow" lies in the range $k = 3 \sim 5$, where the marginal gain rate peaks and time remains moderate.

Table 9: 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 .*

How early path backtracking works. We provide a qualitative example in Table 9 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.

Table 10: 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)	×

How Fibonacci sampling works. Table 10 presents two case studies of automatic classification conducted via Fibonacci sampling. In the war classification example, paths $k = 1$ to $k = 3$ all converge on related but incorrect categories such as "diplomatic initiatives." The correct answer,

"historical wars," only emerges at $k = 8$ with a clear reasoning chain—illustrating the pattern observed in Section 2.1, where early paths often cluster around the same error if the first prediction is wrong. In such cases, Fibonacci sampling helps bypass these local error clusters and reach the correct answer more efficiently.

In the architecture example, where the top-ranked path is already correct, early paths ($k = 1$–3) also yield accurate labels, with $k = 3$ providing an explicit causal explanation. Although Fibonacci may skip some additional correct paths (e.g., $k = 4$, 6, 7), the correct answer remains dominant in aggregated confidence, allowing it to be recovered reliably.

5 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, Synthetic Prompting, 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). Self-Consistency further enhances performance by aggregating diverse reasoning paths (Wang et al., 2022a; Wang & Zhou, 2024). 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). 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 & 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 (e.g., SPIN, SpecEE) 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.

6 Conclusion and Future Work

We propose GCoT-decoding, a general decoding strategy that extends earlier work to broader QA tasks. We refine the branching method for generating candidate paths, which further boosts performance. Experiments show that our method consistently improves the reasoning ability of language models of various sizes and offers greater robustness to task drift.

Beyond the trade-off discussion in the current paper, we are actively exploring optimizations such as early path pruning to reduce computational overhead. At the same time, we plan to extend the evaluation of GCoT-Decoding to a wider range of tasks, particularly those requiring step-by-step reasoning (e.g., structured text generation, logical inference, or multi-hop reasoning), which better align with the strengths of CoT-based methods. For summarization-like tasks where reasoning is less explicit, we will investigate hybrid approaches that selectively apply GCoT-Decoding only to reasoning-intensive components, thereby combining efficiency gains with broader applicability.

Ethics Statement

Our work focuses on developing a general decoding framework (GCoT-Decoding) to enhance the reasoning capabilities of large language models (LLMs). This research does not involve human subjects, sensitive personal data, or proprietary datasets, and all benchmarks used (e.g., GSM8K, MultiArith, SQuAD, BARQA, etc.) are publicly available.

Reproducibility Statement

We have taken multiple measures to ensure reproducibility. All datasets employed are publicly accessible, and preprocessing steps are documented in Appendix C. Algorithmic details, including pseudocode for path sampling, backtracking, and semantic clustering, are provided in Appendix A. Sensitivity experiments on hyperparameters are reported in Appendix B. Experimental settings such as model scales, branching parameters, and evaluation metrics are specified in Section 4.1. Furthermore, we plan to release anonymized source code as supplementary material, enabling independent replication of all experiments.

References

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

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

Wenqing Chen, Weicheng Wang, Zhixuan Chu, Kui Ren, Zibin Zheng, and Zhichao Lu. 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)*, pp. 14162–14167. Association for Computational Linguistics (ACL), 2024.

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

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

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

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

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

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

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

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

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

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

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 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)*, 2023.

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

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

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

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In Lluís Màrquez, Chris Callison-Burch, and Jian Su (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

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

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 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)*, pp. 783–791, 2024.

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

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. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

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

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

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

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

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

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

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

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

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

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

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

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

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

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

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

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

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

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

Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. 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)*, pp. 6642–6650. International Joint Conferences on Artificial Intelligence, 2024.

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

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

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

A 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.

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* $g_i = \mathrm{gen}_2(p_i)$ **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 $s_{i,j^*} \ge \tau$; if none exist, set $j^* = N + 1$;
  if $j^* \le N$ **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.

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.

B SENSITIVITY TO HYPERPARAMETERS

We provide the results of sensitivity experiments on the similarity threshold τ and the confidence threshold δ in Table 11.

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

C PROMPT DEMONSTRATION EXAMPLES

Figure 5 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 5: 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 6 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.

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

D 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.

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

Category	Method	Gemma-7B	Llama-3.1-8B
Clustering (with First-in-Cluster)	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 (with Greedy Clustering)	First-in-Cluster	**54.6**	**67.2**
	Cluster Centroid	47.8	60.4
	Max-Conf	48.2	60.9

E 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 9), the token-level confidence exhibits a sharp local drop, and subsequent tokens mostly elaborate on this misconception rather than correcting it. In these cases, backtracking 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.

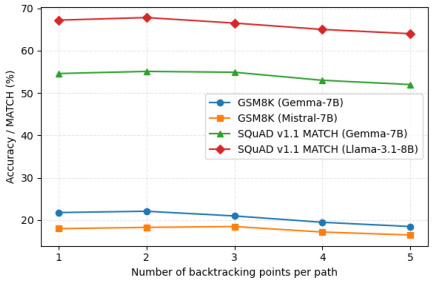


Figure 7: Effect of the maximum number of backtracking points per path under a fixed overall path budget.

Figure 7 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.

F EFFECT OF ANSWER-EXTRACTION TEMPLATES

In Section 2.1, 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.

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

Table 13: Ablation on answer-extraction templates for GCoT-decoding on SQuAD v1.1.

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.

G EMBEDDING MODEL ABLATION FOR SEMANTIC CLUSTERING

In Section 3.3, 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.

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.

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

H 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.

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

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, and often slightly better.

I The Use of Large Language Models

This manuscript used a large language model only for light editorial support—namely grammar and spelling checks, minor language polishing, and table formatting. The LLM did not generate scientific content, results, analyses, or claims. All edits were reviewed by the authors, and the authors remain fully responsible for the final text.