Efficient Latent Semantic Clustering for Scaling Test-Time Computation of LLMs

Sungjae Lee[1], Hoyoung Kim[2], Jeongyeon Hwang[2], Eunhyeok Park[1,2], Jungseul Ok*[1,2],

[1]Department of Computer Science and Engineering, POSTECH, South Korea
[2]Graduate School of Artificial Intelligence, POSTECH, South Korea
{sungjaelee25,cskhy16,jeongyeon.hwang,eh.park,jungseul}@postech.ac.kr

Abstract

Scaling test-time computation, generating and analyzing multiple or sequential outputs for a single input, has become a promising strategy for improving the reliability and quality of large language models (LLMs), as evidenced by advances in uncertainty quantification and multi-step reasoning. A key shared component is semantic clustering, which groups outputs that differ in form but convey the same meaning. Semantic clustering enables estimation of the distribution over the semantics of outputs and helps avoid redundant exploration of reasoning paths. However, existing approaches typically rely on external models, which introduce substantial computational overhead and often fail to capture context-aware semantics. We propose Latent Semantic Clustering (LSC), a lightweight and context-sensitive method that leverages the generator LLM's internal hidden states for clustering, eliminating the need for external models. Our extensive experiments across various LLMs and datasets shows that LSC significantly improves the computational efficiency of test-time scaling while maintaining or exceeding the performance of existing methods.

1 Introduction

Scaling test-time computation has emerged as a promising strategy to improve the reliability and quality of responses from large language models (LLMs) by generating multiple responses for a single input (Kuhn et al., 2023; Lin et al., 2024; Nikitin et al., 2024; Yao et al., 2023; Hao et al., 2023; Snell et al., 2025). For reliability, uncertainty quantification methods assess the semantic diversity among the multiple outputs to estimate model confidence (Kuhn et al., 2023; Lin et al., 2024; Nikitin et al., 2024). For quality, reasoning methods explore and aggregate multiple reasoning paths (Yao et al., 2023; Hao et al., 2023).

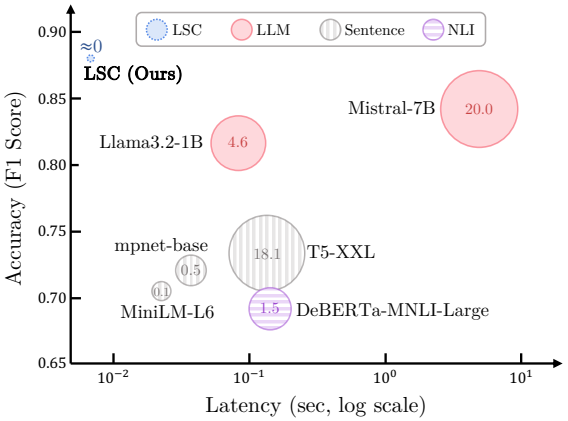


Figure 1: Comparison of LSC and NLI/embedding models in terms of latency (sec), clustering performance (F1 score), and memory usage (GB). F1 score is plotted against log-scaled latency; circle size denotes square-root memory usage. Circles with the same hatch pattern denote the same model type (i.e., LLM, sentence-embedding, or NLI) and numbers inside circles indicate memory usage. LSC achieves the best performance with minimal latency and memory usage. Detailed experimental results are described in Table 3.

A key component in such test-time scaling methods is *semantic clustering*, which groups responses with the same meaning (despite their diverse forms). Indeed, effective clustering can improve uncertainty quantification (Kuhn et al., 2023) by computing the distribution over distinct meanings of responses, and accelerate multi-step reasoning by avoiding unnecessary explorations of (semantically) duplicated reasoning paths (Lee et al., 2025). However, semantic clustering is challenging in open-ended tasks, where the output space is large, ambiguous, and diverse. Reasoning tasks, in particular, are inherently open-ended.

For semantic clustering, existing test-time scaling methods (Kuhn et al., 2023; Qiu and Miikkulainen, 2024) typically employ external models such as natural language inference (NLI) (Nikitin et al., 2024; Lin et al., 2024) or sentence embedding

*Corresponding Author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24126–24144
November 4-9, 2025

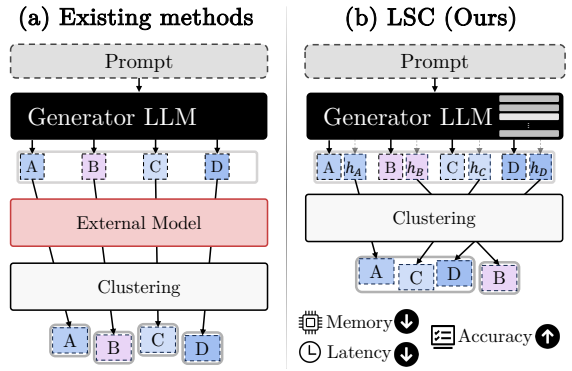


Figure 2: Comparison of semantic clustering frameworks with test-time computation scaling scenario with LLMs. (a) Previous methods rely on external NLI/embedding models to cluster generated sequences. (b) Our latent semantic clustering offers a more efficient and effective alternative by directly leveraging internal hidden representations of LLMs. same colors (*i.e.*, blue and purple) denote semantically equivalent meanings. A concrete example with generated sequences by LLMs is shown in Figure 5.

models (Abdaljalil et al., 2025). Such approaches with external models incur significant computational overhead due to additional inference. In addition, they often fail to capture context-dependent semantics, as they operate separately from the generator of responses. As a result, they are both inefficient and less applicable to test-time scaling methods.

To address these limitations, we propose a lightweight yet effective method for semantic clustering, called Latent Semantic Clustering (LSC). The key idea is to directly leverage the internal latent representations of the generator, eliminating the need for external models (Figure 2). LSC offers two main advantages for test-time scaling as shown in Figure 1. First, it is highly *efficient*—requiring only minimal memory to store hidden vectors and incurring virtually no additional computation. Second, it naturally captures context-aware semantics embedded in the generator's representations, leading to more accurate clustering and effective improvement of test-time scaling methods. Our experiment across various LLMs and datasets validates the efficiency and effectiveness of LSC in uncertainty quantification and multi-step reasoning.

2 Background: Semantic Clustering

This section begins with a formal description of semantic clustering (Section 2.1). We then describe how semantic clustering can be applied to enhance uncertainty quantification (Section 2.2) and to facilitate efficient exploration in LLM reasoning (Section 2.3), both of which commonly leverage test-time computation scaling. The related work is discussed in detail in Appendix A.

2.1 Semantic Clustering with LLMs

To identify semantically consistent or distinct sequences generated by LLMs under test-time scaling strategies, semantic clustering groups outputs based on their semantics rather than lexical form. Specifically, we generate a set of sequences $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$ by sampling from the generator LLM $\mathcal{M}$ given a context x. Here, the context x may include task instructions, in-context examples, the input problem or question, and, when applicable, intermediate reasoning trajectories from earlier steps. To identify the semantic relationships among the generated sequences, the most widely used approach (Kuhn et al., 2023) employs bi-directional classification using an NLI model. It evaluates both $\theta_{ext}(\mathbf{s}_i, \mathbf{s}_j)$ and $\theta_{ext}(\mathbf{s}_j, \mathbf{s}_i)$, where θ_{ext} denotes an external model that determines whether one sequence semantically entails the other. If both classification results are *entailment*, the two sequences are grouped into a semantic cluster $c \in \mathcal{C}$, which is called bi-directional entailment clustering (BDEC) proposed for LLM uncertainty quantification. Recently, subsequent works (Abdaljalil et al., 2025) have replaced θ_{ext} with embedding models such as BERT (Devlin et al., 2019) and Sentence-BERT (Reimers and Gurevych, 2019), applying clustering algorithms to the embeddings.

2.2 Semantic Uncertainty

In open-ended natural language tasks, semantic clustering can be utilized to measure the uncertainty in the N texts generated by an LLM for a given context x. Let $\mathcal{C}$ be the set of semantic clusters over the generated sequences $\mathbf{s}_1, ..., \mathbf{s}_N$. The cluster probability $p(c \mid x, \mathcal{S})$ is defined by summing the probabilities of sequences assigned to cluster $c \in \mathcal{C}$ as follows:

$$p(c \mid x, \mathcal{S}) := \sum_{\mathbf{s} \in \mathcal{S}} 1[\mathbf{s} \in c] \cdot p(\mathbf{s} \mid x) \ , \quad (1)$$

where $p(\mathbf{s} \mid x)$ is either computed from token-level generation probabilities as $\prod_i p(s^i \mid s^{<i}, x)$ or approximated as $1/N$.

Based on this distribution, the semantic entropy (Kuhn et al., 2023) on generated sequences

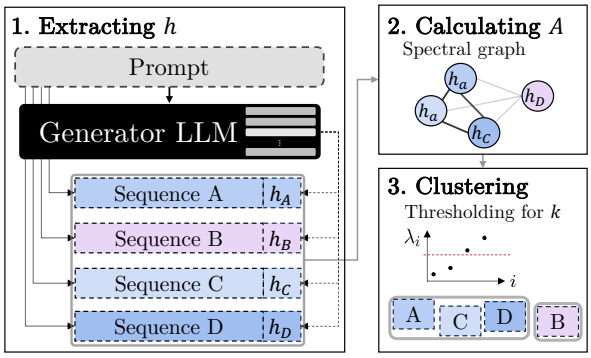


Figure 3: Overview of Latent Semantic Clustering (LSC). LSC consists of three main steps: (1) Extract hidden states during generations, (2) compute pairwise similarities to form an adjacency matrix, and (3) apply spectral clustering while determining the number of clusters k.

$\mathcal{S}$ is defined as:

$$\mathrm{SE}(\mathcal{S}; x, \mathcal{C}) := -\sum_{c \in \mathcal{C}} p(c \mid x, \mathcal{S}) \log p(c \mid x, \mathcal{S}). \quad (2)$$

When generator LLMs produce diverse meanings from a single prompt, semantic entropy is high; when meanings converge, entropy is low, serving as a measure of LLM uncertainty. Here, the reliability of semantic uncertainty estimation hinges on the quality of the clusters $\mathcal{C}$, which are typically derived using external models.

2.3 Semantic Exploration

Beyond linear reasoning methods (Wei et al., 2022; Wang et al., 2022), tree search-based methods expand and explore intermediate reasoning trajectories to enhance the reasoning capabilities of LLMs (Hao et al., 2023; Yao et al., 2023) as test-time scaling strategies. However, these methods often incur significant computational overhead due to redundant exploration of semantically equivalent paths. To alleviate this, recent studies have leveraged semantic clustering to reduce the number of expansions from all d candidate nodes generated by the LLM at each step to only $d' < d$ semantically distinct nodes, thereby enabling more efficient exploration on reasoning trees (Lee et al., 2025; Wang et al., 2025a).

Nonetheless, these methods still rely on external models while demanding additional computational resources and often struggle to effectively incorporate contextual information during reasoning. Notably, Lee et al. (2025) constructed the input of NLI model using only the generated sequences without context, while Wang et al. (2025a) performed additional training on external embedding model due to its limited capacity to effectively handle contextual information as described in Figure 5 and Table 3. In contrast, we perform clustering without external resources by directly utilizing the internal semantic representations of the generator LLMs.

Algorithm 1 Latent Semantic Clustering

Require: Input x, LLM $\mathcal{M}$, threshold τ
Ensure: Set of clusters $\mathcal{C}$

1: During generation of N sequences $\mathcal{S}$ with $\mathcal{M}(x)$, obtain hidden states h_n for each $s \in \mathcal{S}$
2: Construct adjacency matrix A using cosine similarity between hidden states via (6)
3: Determine the number of clusters k by thresholding the eigenvalues of (7) at τ
4: Obtain the cluster set $\mathcal{C}$ of size k by applying spectral clustering on A

3 Method

For efficient semantic clustering, we propose Latent Semantic Clustering (LSC) that captures context-aware semantics by leveraging the hidden states produced by the generator LLMs. Our approach involves three key steps: (1) we extract the hidden states from the generator LLM during inference; (2) we construct an adjacency matrix that maps pairwise semantic similarity between the generated sequences; (3) we perform spectral clustering while determining the optimal number of clusters. The overall procedure is summarized in Figure 3 and Algorithm 1.

Extracting hidden states The first step of LSC extracts latent representations that capture the semantic content of each generated sequence. To do so, we leverage the hidden states from an intermediate Transformer layer during the auto-regressive generation process for i-th token, defined as:

$$h^{i,0} = e^i + p^i \quad (3)$$

$$h^{i,\ell} = \mathrm{TransformerBlock}^{(\ell)}(h^{i,\ell-1}) \quad (4)$$

where e^i is the token embedding, p^i is the positional encoding, and $h^{i,\ell}$ denotes the hidden state at the ℓ-th Transformer layer. Then, the generator LLM $\mathcal{M}$ can be defined as:

$$\mathcal{M}(s^{i+1} \mid s^{\leq i}) = \mathrm{softmax}(W h^{i,L}), \quad (5)$$

where $h^{i,L}$ is the last layer's hidden state at position i, and W is the parameter of the LLM head.

We extract the hidden state $h^{i,\ell}$ from a predefined l-th Transformer layer at the last generated token for each generated sequence, and denote it as h_n for the n-th generated sequence for simplicity. In other words, we store only a single hidden state per sequence, regardless of the number of tokens within the sequence. This hidden state h_n serves as a representation of $\mathbf{s}_n$, and we repeat this process across N samplings of the generator LLM $\mathcal{M}$ as part of test-time computation scaling.

Constructing adjacency matrix Given a set of hidden states $\{h_1, h_2, \ldots, h_N\}$ for each generated sequence $\mathbf{s}_n$, we compute the pairwise cosine similarity between each pair of hidden states to measure semantic similarity for subsequent clustering. Specifically, the similarity between the m-th and n-th hidden states is defined as:

$$a_{m,n} = \frac{h_m \cdot h_n}{\|h_m\| \, \|h_n\|} , \tag{6}$$

and $a_{m,n}$ construct the adjacency matrix $A \in \mathbb{R}^{N \times N}$. We observe that the cosine similarity $a_{m,n}$ between hidden states can effectively replace NLI-based semantic similarity used in existing soft clustering methods (Lin et al., 2024; Nikitin et al., 2024), as shown in Section 5.

Spectral clustering Given the resulting adjacency matrix $A \in \mathbb{R}^{N \times N}$, we apply spectral clustering to identify semantic clusters among the generated sequences. First, we compute the symmetric normalized Laplacian as follows:

$$L \coloneqq I - D^{-1/2} A D^{-1/2} , \tag{7}$$

where D is the diagonal degree matrix with entries $D_{m,m} \coloneqq \sum_{n=1}^{N} a_{m,n}$. We then calculate the eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ and corresponding eigenvectors $v_1, v_2, \ldots, v_N$ of L. The number of clusters k is determined by counting the number of eigenvalues below a predefined threshold τ, i.e., $k \coloneqq |\{\lambda_i \mid \lambda_i < \tau\}|$, based on the following theoretical result:

Theorem 1. *(Von Luxburg, 2007) The multiplicity of the eigenvalue 0 of L is equal to the number of connected components in the graph defined by the adjacency matrix A.*

In other words, when the adjacency matrix A is binary, the number of connected components—and thus the number of clusters—exactly equals the multiplicity of the zero eigenvalue. For continuous-valued A, the graph typically forms a single connected component; however, the distribution of small eigenvalues of L can still be used to estimate the number of semantically distinct meanings among the generated sequences.

Lastly, we perform hard semantic clustering over generated sequences, facilitating their direct integration into existing methods for uncertainty quantification (Kuhn et al., 2023) and multi-step reasoning (Lee et al., 2025). Specifically, we construct a spectral embedding using the first k eigenvectors $v_1, \ldots, v_k$, and apply k-means clustering in this reduced space to assign each sequence $\mathbf{s}_n$ to a semantic cluster c. In contrast, soft clustering methods leverage elements in spectral clustering such as the eigenvalue distribution to quantify LLM uncertainty (Lin et al., 2024).

4 Experiments

In this section, we empirically evaluate whether the proposed LSC effectively leverages the outputs generated with scaled inference-time computation to improve performance and efficiency across various downstream LLM tasks. To assess the robustness of LSC, we conduct experiments with multiple LLMs, including Llama3-8B-Instruct (Grattafiori et al., 2024) and Mistral-7B-Instruct-v0.1 (Jiang et al., 2023). Additional results are presented in Appendix C. In what follows, we present results on uncertainty quantification tasks (Section 4.1) and multi-step reasoning tasks (Section 4.2). In both tasks, we use the hidden state from an intermediate LLM layer, extracted at the last token of each generated sequence in LSC.

4.1 Uncertainty Quantification

We evaluate the effectiveness of LSC in uncertainty quantification tasks that estimate LLM confidence by analyzing multiple outputs generated from a single input.

Baselines We compare our method to a variety of existing approaches for uncertainty quantification. P(True) (Kadavath et al., 2022) estimates the probability of correctness by prompting additional query to the LLM, while PE (Malinin and Gales, 2020) computes predictive entropy via Monte Carlo dropout without considering inter-sequence semantics. Recent methods can be categorized into soft and hard semantic clustering approaches. Soft clustering methods such as KLE (Nikitin et al., 2024), Deg, EigV, and ECC (Lin et al., 2024) perform spectral clustering using semantic similarity graphs constructed with external NLI models. Specifically,

Method	External Model θ_{ext}	BioASQ		SQuAD		TriviaQA	
		AUROC ↑	AUARC ↑	AUROC ↑	AUARC ↑	AUROC ↑	AUARC ↑
P(True) (Kadavath et al., 2022)	LLM	0.7565	0.6663	0.6744	0.3559	0.7781	**0.7926**
PE (Malinin and Gales, 2020)	-	0.8102	0.7006	<u>0.7413</u>	<u>0.3954</u>	<u>0.7839</u>	<u>0.7912</u>
KLE (Nikitin et al., 2024)	NLI	0.8361	0.6960	0.7199	0.3891	0.7771	0.7755
Deg (Lin et al., 2024)	NLI	0.7740	0.6411	0.6780	0.3466	0.6343	0.7083
EigV (Lin et al., 2024)	NLI	0.5646	0.5198	0.5877	0.2943	0.6210	0.7010
ECC (Lin et al., 2024)	NLI	0.8238	0.7060	0.6783	0.3694	0.7150	0.7677
NumSets (Kuhn et al., 2023)	NLI	0.8318	0.6794	0.7311	0.3751	0.7717	0.7700
SE (Kuhn et al., 2023)	NLI	0.8232	0.6873	0.7148	0.3876	0.7554	0.7666
DSE (Kuhn et al., 2023)	NLI	0.8338	0.6883	0.7288	0.3835	0.7783	0.7752
SE-LSC (Ours)	-	<u>0.8602</u>	**0.7222**	0.7401	0.3941	0.7745	0.7867
DSE-LSC (Ours)	-	**0.8654**	<u>0.7152</u>	**0.7616**	**0.3981**	**0.7843**	0.7855

Table 1: Comparison of the effectiveness of uncertainty quantification methods in terms of AUROC and AUARC on three datasets using Llama3-8B-Instruct. **Bold** and <u>underlined</u> values indicate the best and second-best performance in each setting, respectively.

KLE applies kernel-based entropy on semantic similarity graphs, Deg and EigV use spectral clustering based on graph degrees or eigenvalues, ECC leverages semantic similarity graphs and applies spectral methods based on graph Laplacian embeddings, estimating uncertainty by measuring average distance from the embedding center. In contrast, hard clustering methods such as NumSets, SE, and DSE (Kuhn et al., 2023) group semantically equivalent sequences by identifying pairwise entailment relations through external NLI models. NumSets uses the number of clusters constructed as an uncertainty measure, SE measures predictive entropy over semantic equivalence classes built from NLI-based bidirectional entailment clustering, and DSE is a discrete variant of SE that does not rely on output probabilities. However, our methods, SE-LSC and DSE-LSC, replace the external NLI models in SE and DSE with latent hidden states from the LLM, enabling more efficient and effective uncertainty estimation. We further extend this substitution to soft clustering methods in Section 5.

Datasets We evaluate our method on three question answering (QA) datasets, each representing a distinct domain. TriviaQA (Joshi et al., 2017) reflects the open-domain setting with diverse and noisy trivia questions collected from the web. SQuAD (Rajpurkar et al., 2016) represents the general-domain setting, featuring clean and well-structured question–context pairs from Wikipedia. BioASQ (Krithara et al., 2023) corresponds to the biomedical domain, consisting of factoid questions that require expert-level understanding. We follow the experimental setup used in Nikitin et al. (2024).

Results Table 1 compares various uncertainty quantification methods using the Area Under the Receiver Operating Characteristic (AUROC) and the Area Under the Accuracy-Rejection Curve (AUARC). AUROC evaluates how accurately a method ranks predictions by computed uncertainty, assigning higher uncertainty to incorrect answers. In contrast, AUARC reflects calibration by tracking how accuracy improves as high-uncertainty predictions are progressively removed based on a threshold. While existing NLI-based baselines demonstrate moderate performance, our proposed methods, SE-LSC and DSE-LSC, consistently achieve the highest AUROC scores across all settings and yield the best or comparable AUARC values. Notably, our methods outperform or match existing baselines without requiring additional computational costs such as NLI models. These results suggest our method provides more reliable uncertainty estimation in uncertainty quantification tasks. Additional results with more LLMs are reported in Appendix C.

4.2 LLM Reasoning

To evaluate the effectiveness and efficiency of LSC in enhancing reasoning capabilities, we investigate multi-step reasoning tasks that use tree search as a test-time computation scaling strategy, where LSC is applied to merge semantically redundant reasoning paths. Specifically, tree search-based reasoning approaches explore multiple intermediate reasoning paths using algorithms such as beam search and Monte Carlo tree search (MCTS).

Baselines We consider three baseline methods for tree search-based reasoning. Tree-of-

Benchmark	Method	External Model θ_{ext}	Llama3-8B-Instruct			Mistral-7B-Instruct		
			Accuracy ↑	# of LLM inferences ↓	# of NLI inferences ↓	Accuracy ↑	# of LLM inferences ↓	# of NLI inferences ↓
GSM8K	ToT (Yao et al., 2023)	-	0.79	104.80	-	0.61	122.16	-
	RAP (Hao et al., 2023)	-	0.83	128.40	-	0.67	161.86	-
	SExp (Lee et al., 2025)	NLI	0.85	82.63	62.86	**0.68**	98.79	78.26
	SExp-LSC (Ours)	-	**0.86**	**81.40**	-	0.67	**84.38**	-
ARC	ToT	-	0.80	149.59	-	0.62	221.36	-
	RAP	-	0.81	196.96	-	**0.71**	313.20	-
	SExp	NLI	**0.83**	96.32	70.42	**0.71**	155.13	121.67
	SExp-LSC (Ours)	-	**0.83**	**82.97**	-	**0.71**	**120.87**	-

Table 2: Comparison of accuracy and inference efficiency for reasoning methods on both GSM8K and ARC datasets using Llama3-8B-Instruct and Mistral-7B-Instruct. **Bold** values denote the best results, and underlined values indicate the second-best.

Clustering	Model Type	Model Name	Context	Memory (GB) ↓	Latency (sec) ↓	F1 Score ↑	Precision ↑	Recall ↑
BDEC	NLI	DeBERTa-MNLI-Large	✗	1.5212	0.1298	0.4365	**0.9558**	0.3865
			✓		0.1447	0.6904	0.6232	**0.9442**
Spectral Clustering	Sentence BERT	all-MiniLM-L6-v2	✗	0.0926	0.0225	0.7058	0.7414	0.8070
			✓		0.0217	0.6888	0.6603	0.8816
		all-mpnet-base-v2	✗	0.4618	0.0372	0.7198	0.7043	0.8703
			✓		0.0361	0.7078	0.6653	0.9101
		gtr-t5-xxl	✗	18.1341	0.1358	0.7333	0.7407	0.8468
			✓		0.3287	0.7150	0.6497	0.9353
	Decoder-only LLM	Llama3.2-1B	✗	4.6127	0.0497	0.7098	0.7234	0.8246
			✓		0.0820	0.8160	0.8369	0.8805
		Mistral-7B	✗	19.9969	4.4361	0.6998	0.6812	0.8774
			✓		4.9110	0.8415	0.8816	0.8844
		LSC	✓	$\approx$ **0**	**0.0069**	**0.8789**	0.8825	0.9367

Table 3: Comparison of clustering performance for NLI, sentence-embedding, and LLM-based models in multi-step reasoning. We present F1 score, precision, recall, latency (sec), and memory usage (GB) to evaluate both clustering quality and computational efficiency. **Bold** values denote the best results, and underlined values indicate the second-best.

Thoughts (ToT) (Yao et al., 2023) and Reasoning-via-Planning (RAP) (Hao et al., 2023) are well-known approaches that leverage beam search and MCTS, respectively. Semantic Exploration (SExp) (Lee et al., 2025) is a more recent work that improves tree search efficiency by adopting semantic clustering to avoid semantically redundant exploration on reasoning trees. We note that SExp utilizes the BDEC algorithm (Kuhn et al., 2023) with an external NLI model, DeBERTa-MNLI-Large (He et al., 2020), for semantic clustering.

Datasets To evaluate the reasoning capabilities of LLMs, we use two benchmarks: GSM8K (Cobbe et al., 2021), a dataset consisting of 8.5k math word problems that require multi-step mathematical reasoning, and the AI2 Reasoning Challenge (ARC), which contains 7.8k science questions derived from grade-school science exames. Following the evaluation setup of SExp (Lee et al., 2025), we randomly sample 400 examples from the test sets of GSM8K and ARC for evaluation.

Results In Table 2, we evaluate the effect of LSC on multi-step reasoning tasks. ToT and RAP require a large number of LLM inferences, as they expand and explore semantically duplicate reasoning paths. SExp reduces this computational cost by merging semantically redundant paths using an external NLI model, but still relies on external supervision. In contrast, our SExp-LSC achieves comparable accuracy without any external model by leveraging internal LLM representations. Notably, on the ARC dataset, SExp-LSC reduces the number of LLM inferences by 22.58% while maintaining accuracy, demonstrating the efficiency and effectiveness of our approach. This reduction in LLM inferences stems from LSC's strong clustering performance, enabled by its use of context-aware semantic repre-

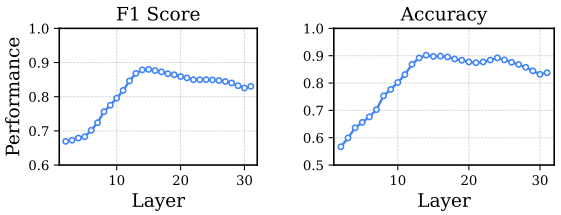


Figure 4: Layer-wise clustering performance using Llama3-8B-Instruct hidden states. Intermediate and later layers demonstrate higher F1 scores and accuracy than earlier layers.

Layer selection	AUROC	AUARC
16th only	0.8654	0.7152
16th & 8th	**0.8680**	**0.7157**
16th & 24th	0.8649	0.7139
16th & 32nd	0.8639	0.7136

Table 4: Comparison of layer selection strategies for LSC on the BioASQ dataset using Llama3-8B-Instruct. Results are reported in terms of AUROC and AUARC. **Bold** values denote the best results, and underlined values indicate the second-best

sentations to merge semantically identical reasoning paths.

5 Further Analyses

Effect of LSC on clustering To evaluate the effectiveness and efficiency of leveraging hidden states from the generator LLM for semantic clustering, we compare LSC with existing NLI- and embedding-based approaches. Specifically, we evaluate clustering performance on the first-level sequences of reasoning trees generated by Llama3-8B-Instruct on GSM8K. To construct clustering labels, we use Llama3-70B-Instruct to generate pairwise semantic equivalence labels by using a prompt, as illustrated in Figure A5. Clustering performance is measured by computing precision, recall, and F1 score for the generated sequences of each example and averaging across all cases. We also assess computational efficiency by measuring memory usage and latency. To evaluate how well each model handles contextual information, we conduct experiments both with and without including the context as part of the input.

NLI-based models show a trade-off between precision and recall depending on whether context is included, while BERT-based embeddings struggle to capture context-dependent semantics, resulting in suboptimal clustering as demonstrated in Figure 1 and Table 3. For external models, we measured the memory consumption of both the models and their extracted embeddings. In contrast, LLM-based embeddings benefit significantly from context: both precision and recall increase when contextual information is included. Our LSC method, which leverages internal LLM representations, achieves the highest F1 score with negligible memory and latency overhead, maintaining semantic fidelity while offering superior efficiency. Note that LSC requires only $\approx$ **0.000061** GB of memory to store LLM hidden states, sized as the number of generated sentences multiplied by the hidden state dimension.

Layer selection To better understand the layer-wise effectiveness of LLMs, we find that intermediate and later layers yield robust and effective semantic representations (Figure 4), in line with prior findings (Skean et al., 2025). In addition, we observe performance gains when expanding the embedding dimensions for clustering. Table 4 reports the results of DSE-LSC implemented by concatenating the hidden states from two layers for the last token of each generated sequence. Specifically, we compare using only the 16th (intermediate) layer of the 32 Transformer layers in Llama3-8B-Instruct against combining it with other layers in uncertainty quantification on the BioASQ dataset. The results show that incorporating the hidden state from the 8th (first-quarter) layer alongside the intermediate layer yields slight improvements over using the intermediate layer alone. We expect that leveraging not only multiple layers or token-level embeddings, but also attention weights, could be a promising direction, which we leave for future work.

Context-aware semantic clustering To qualitatively assess whether LSC captures context-aware semantics, Figure 5 presents an example from GSM8K using Llama3-8B-Instruct. The figure illustrates the first level of reasoning trees generated by SE and SE-LSC. When the shared context x is excluded from the input, the NLI-based clustering model in SE lacks context awareness and instead relies solely on surface-level sentence semantics, resulting in all sequences being grouped into separate clusters. Indeed, in the absence of shared context, sequences A, C, and D appear to convey distinct meanings, despite being semantically equivalent when the context is considered. However, even when the context is included, the model still fails to accurately identify and cluster semantically equiva-

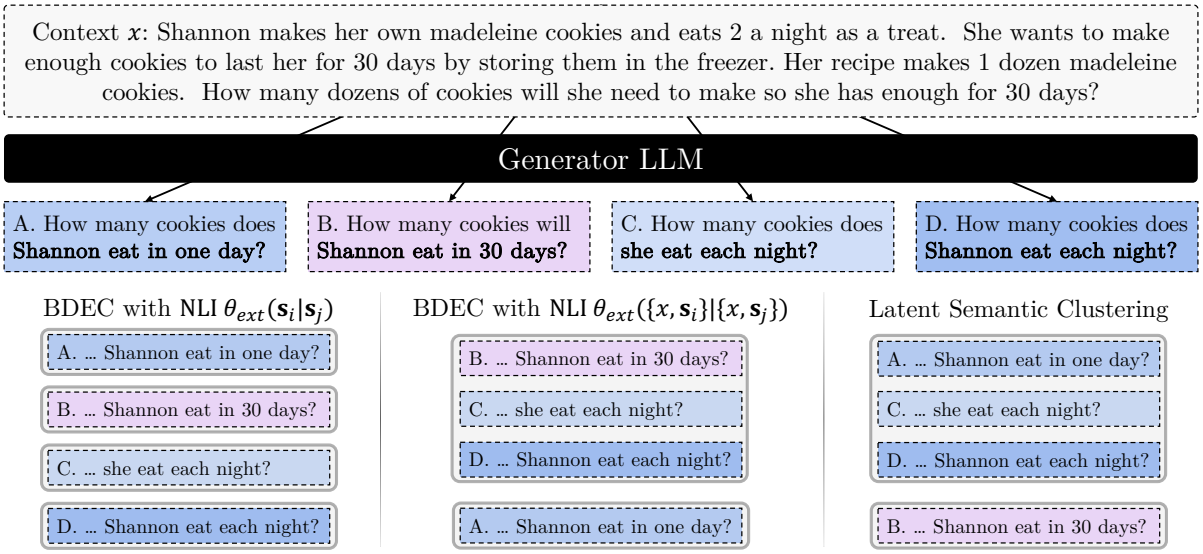


Figure 5: Example of semantic clustering from GSM8K using Llama3-8B-Instruct comparing SExp and SExp-LSC. The NLI-based model fails to group context-dependent semantically identical sequences (e.g., A, C and D), while LSC, using context-conditioned hidden representations, correctly clusters semantically equivalent sequences under a given context. This highlights a key limitation of NLI-based methods: without context, they miss context-dependent meaning, and with context, they often fail to properly process it along with the generated sequences. The full constructed reasoning trees are shown in Figure A1.

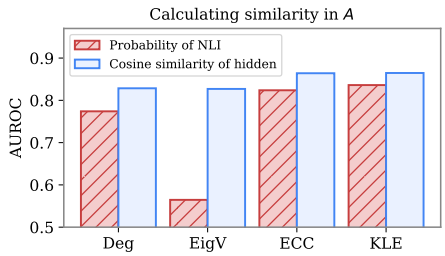


Figure 6: Comparison of different adjacency matrix construction methods for soft clustering-based uncertainty quantification approaches leveraging semantic similarities between generated sequences.

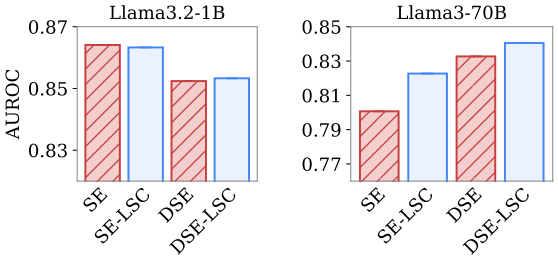


Figure 7: Effect of LLM size on the effectiveness of LSC. Comparison of AUROC scores for SE, DSE, SE-LSC, and DSE-LSC on the BioASQ dataset using Llama3.2-1B-Instruct and Llama3-70B-Instruct.

lent sentences, as it does not effectively incorporate contextual information. In contrast, LSC successfully performs context-aware semantic clustering by leveraging the hidden representations from the generator LLM, which are inherently conditioned on the input context.

Semantic similarity as soft edge Subsequent works of Kuhn et al. (2023) consider semantic similarity, rather than strict equivalence, for uncertainty quantification. KLE (Nikitin et al., 2024), Deg, EigV, and ECC (Lin et al., 2024) leverage spectral clustering based on an adjacency matrix constructed using an NLI model, enabling uncertainty estimation without hard semantic clustering. Therefore, their adjacency matrix can be substituted with our adjacency matrix in Equation (6), which is based on cosine similarity between hidden states. We evaluate these methods to compare adjacency matrix construction based on NLI with our approach on the BioASQ dataset using Llama3-8B-Instruct. As shown in Figure 6, our method described in Equation (6) achieves superior performance in both AUROC and AUARC, suggesting the potential of computing semantic similarity from hidden states of the generator LLMs.

Smaller and larger LLMs To examine the robustness of LSC with respect to LLM size, we conduct experiments using Llama3.2-1B-Instruct (MetaAI, 2024) and Llama3-70B-Instruct (Grattafiori et al., 2024) on the BioASQ dataset, performing uncertainty quantification. For the smaller LLM (*i.e.,* Llama3.2-1B-Instruct), SE-LSC and DSE-LSC shows comparable performance in AUROC and AUARC to the SE and DSE as

Clustering method	Precision ↑	Recall ↑	F1 Score ↑
k-means	0.8865	0.7232	0.7387
AHC	**0.9308**	0.8837	0.8664
DBSCAN	0.8383	**0.9660**	0.8676
LSC (Ours)	0.8825	0.9367	**0.8789**

Table 5: Comparison of clustering methods, k-means, AHC, DBSCAN, and LSC, using Llama3-8B-Instruct hidden states for semantic clustering on the GSM8K dataset. We report precision, recall, and F1 score. **Bold** values denote the best results, and underlined values indicate the second-best.

shown in Figure 7. In contrast, for the larger LLM (*i.e.,* Llama3-70B-Instruct), SE-LSC and DSE-LSC shows greater improvements over the baselines, SE and DSE. These results suggest that LSC may be more effective with larger LLMs.

Different clustering algorithms We compare LSC with standard clustering algorithms, including k-means, Agglomerative Hierarchical Clustering (AHC), and DBSCAN, using the same hidden states from Llama3-8B-Instruct. For all baselines, we report the best F1 score obtained by hyperparameter search, except for k-means, where the number of clusters is determined using the elbow method. As shown in Table 5, LSC achieves the highest F1 score of 0.8789, outperforming other clustering methods and indicating its practical effectiveness for semantic clustering in downstream tasks.

6 Conclusion

In this work, we propose Latent Semantic Clustering (LSC), a lightweight and context-aware approach for identifying semantically equivalent outputs generated by LLMs for test-time computation scaling. Rather than relying on external models as in prior works, LSC leverages the generator LLM's internal hidden states to perform context-aware semantic clustering with minimal overhead. Our experiments show that LSC achieves strong performance on key tasks like uncertainty quantification and multi-step reasoning, while substantially reducing computational cost. LSC consistently outperforms prior semantic clustering methods in both clustering quality and computational efficiency. We believe LSC offers a practical and scalable solution, applicable to any white-box LLMs, for enhancing the reliability and quality of LLM responses, thereby improving test-time computation scaling strategies.

Limitations

While our approach demonstrates strong performance and efficiency, there are limitations that suggest directions for future improvement. First, our method relies on the hidden state from a single intermediate layer and the last token, which provides an efficient and effective representation in practice. However, incorporating multiple layers, aggregating token-level information, or utilizing attention patterns may further enhance representation quality and is a promising direction for future work. Second, LSC requires access to the generator model's hidden states, and thus is currently applicable only to white-box LLMs. While this design enables improvements in efficiency and effectiveness, it may limit applicability in scenarios involving black-box models. Exploring alternatives that approximate internal representations without direct access could help broaden the scope of LSC.

Acknowledgements

This work was supported by the Korea Institute for Advancement of Technology (KAIT), funded by the Ministry of Trade, Industry and Energy (MOTIE), Republic of Korea (RS-2025-00564342), the Seoul Business Agency (SBA) funded by The Seoul Metropolitan Government (SP240008, Seoul R&BD Program), and the Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH); RS-2024-00436680, Global Research Support Program in the Digital Field program). And, this project was also supported by Microsoft Research Asia.

References

Samir Abdaljalil, Hasan Kurban, Parichit Sharma, Erchin Serpedin, and Rachad Atat. 2025. Sindex: Semantic inconsistency index for hallucination detection in llms. *arXiv preprint arXiv:2503.05980*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep

bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Jiatong Han, Jannik Kossen, Muhammed Razzak, and Yarin Gal. 2024. Semantic entropy neurons: Encoding semantic uncertainty in the latent space of llms. In *MINT: Foundation Model Interventions*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927*.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Sungjae Lee, Hyejin Park, Jaechang Kim, and Jungseul Ok. 2025. Semantic exploration with adaptive gating for efficient problem solving with language models. *Association for Computational Linguistics*.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*.

Andrey Malinin and Mark Gales. 2020. Uncertainty estimation in autoregressive structured prediction. *arXiv preprint arXiv:2002.07650*.

MetaAI. 2024. Llama 3.2-1b-instruct. https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct. Instruction-tuned multilingual language model.

Alexander Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. 2024. Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities. *Advances in Neural Information Processing Systems*, 37:8901–8929.

Alina Petukhova, João P Matos-Carvalho, and Nuno Fachada. 2025. Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108.

Xin Qiu and Risto Miikkulainen. 2024. Semantic density: Uncertainty quantification for large language models through confidence measurement in semantic space. *Advances in neural information processing systems*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. *International conference on machine learning*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling llm test-time compute optimally can be more effective than scaling model parameters.

The Eleventh International Conference on Learning Representations.

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Ante Wang, Linfeng Song, Ye Tian, Dian Yu, Haitao Mi, Xiangyu Duan, Zhaopeng Tu, Jinsong Su, and Dong Yu. 2025a. Don't get lost in the trees: Streamlining llm reasoning by overcoming tree search exploration pitfalls. *arXiv preprint arXiv:2502.11183*.

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

Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, and Rui Wang. 2025b. Latent space chain-of-embedding enables output-free LLM self-evaluation. In *The Thirteenth International Conference on Learning Representations*.

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

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *The Eleventh International Conference on Learning Representations*.

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. *Advances in neural information processing systems*, 36:11809–11822.

Appendix

A Related Work

Scaling LLM test-time computation Leveraging increased computation at inference time-through parallel or sequential sampling-allows LLMs to enhance the quality and reliability of their outputs on various NLP tasks (Wei et al., 2022; Yao et al., 2023; Snell et al., 2025). A simple yet powerful approach generates multiple outputs for a given prompt and identifies the most consistent one among the outputs (Wang et al., 2022; Xiong et al., 2024). Recently, several studies have proposed incorporating semantic relationship to identify significant outputs in open-ended natural language generation tasks by considering semantics. For example, Kuhn et al. (2023) generates multiple sequences and clusters them to estimate LLM uncertainty. Similarly, studies on LLM reasoning identify semantically equivalent sequences among multiple samples to avoid redundant reasoning paths and encourage exploration of more significant ones (Lee et al., 2025; Wang et al., 2025a).

Semantic clustering in LLMs. Existing methods for semantic clustering typically have relied on external models to evaluate semantic relationship between textual sequences. For example, Kuhn et al. (2023) employ a natural language inference (NLI) model to identify these relationships and performs clustering based on the model's outputs. Subsequent works also leverage the NLI models and adopt spectral clustering to quantify uncertainty (Lin et al., 2024; Nikitin et al., 2024; Qiu and Miikkulainen, 2024). Alternatively, Abdaljalil et al. (2025) utilize Sentence-BERT-based embedding model and perform clustering based on cosine similarity between the embeddings. However, these approaches rely on external models and incur additional computational overhead, which can hinder scalability and efficiency. In addition, the external models such as NLI and embedding models often struggle to effectively handle model inputs consisting of context and generated sequences. Notably, prior studies focus solely on uncertainty quantification, overlooking the extension to different LLM tasks.

Leveraging hidden states in LLMs Recent studies have explored the utility of LLM hidden states across various downstream tasks. In text clustering, embeddings derived from LLMs exhibit rich semantic information, outperforming conventional sentence embeddings in capturing contextual similarity (Petukhova et al., 2025). However, this work focuses on clustering a given set of input texts, rather than analyzing internal representations across generated outputs. Other lines of work use hidden states to estimate uncertainty for hallucination detection (Kossen et al., 2024; Han et al., 2024), but often rely on training additional lightweight multi-layer perceptrons on top of the hidden states, which introduces computational overhead. More recently, several studies have shown that LLMs can reason and self-evaluate directly in the latent space, without producing explicit outputs (Hao et al., 2024; Wang et al., 2025b). However, we instead leverage latent hidden states for clustering generated outputs, enabling applications to test-time scaling tasks without additional training or supervision.

B Experimental Settings

The hyper-parameters used in our experiments are listed in Table A1 for the uncertainty quantification tasks. The clustering threshold τ for LSC is selected based on performance on a validation set of 100 examples per task. Table A2 lists the hyper-parameters used for multi-step reasoning tasks.

C Extended Experimental Results

Table A3 presents extended experimental results for the uncertainty quantification task, evaluated using three datasets across two models: Llama3-8B-Instruct and Mistral-7B-Instruct.

D Prompt

Figure A2 illustrates the 5-shot in-context prompt used for uncertainty quantification. Figures A3 and A4 show the 1-shot in-context prompts used for semantic exploration. These are identical to the prompts used in SExp (Lee et al., 2025) experiments.

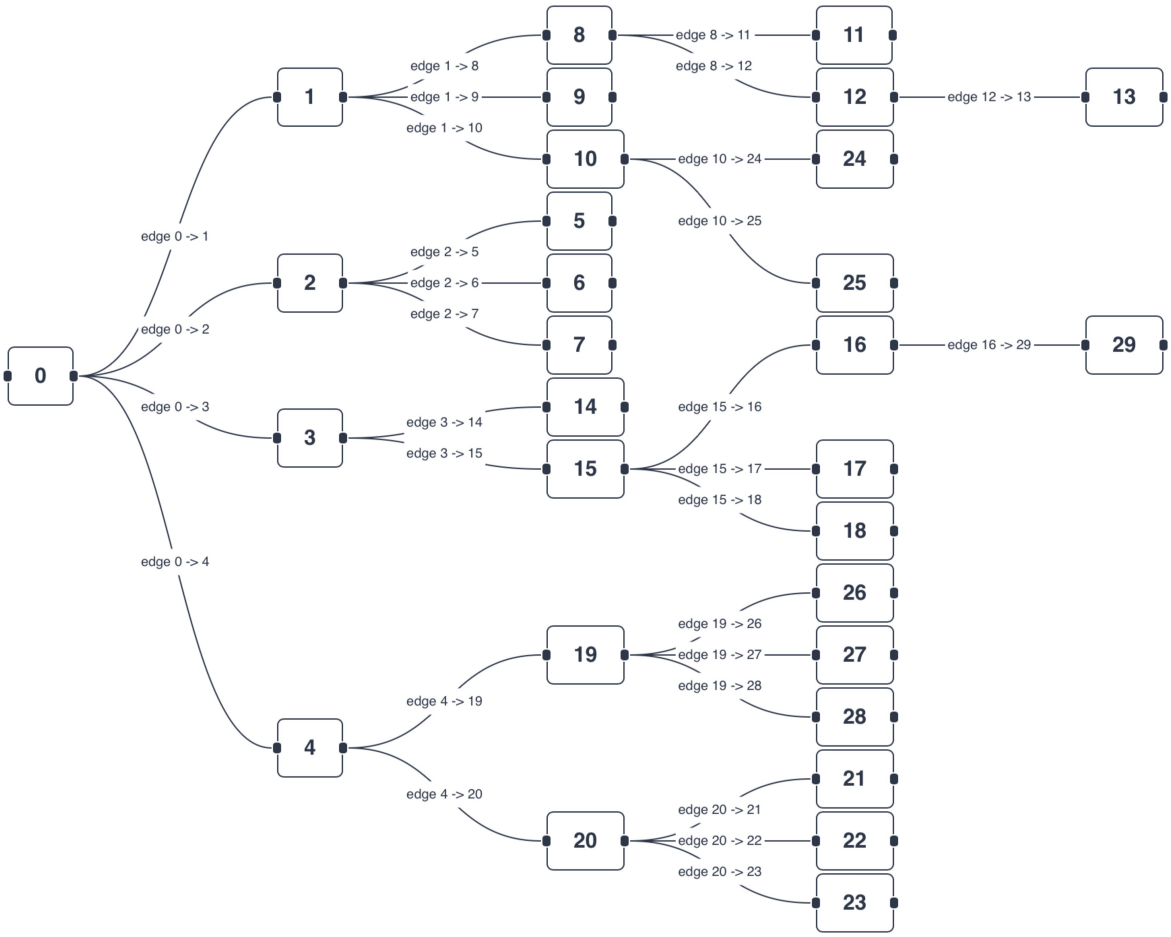
Setting	Hyperparameter	Value
LLM-related setting	temperature	1.0
	top-k	50
	top-p	1.0
General setting	batch size	1
	the number of examples (prompts)	1
Deg, EigV, ECC	threshold for eigenvalues	*
Ours (LSC)	threshold for clustering	*

Table A1: The default hyperparameters used in our experiments are categorized into LLM-related, general, and method-specific settings. * indicates that the hyperparameter is selected based on the best threshold determined from the validation set.

Setting	Parameter	Value
LLM-related setting	temperature	0.8
	top-k	50
	top-p	0.95
General setting	batch size	1
	the number of examples (prompts)	1
MCTS setting	depth limit	5
	the number of iterations	10
	the number of actions	4
	reward alpha	0.5
	the number of confidences	8
	a default value of reward confidence	0.8
ToT setting	beam size	3
	depth limit	5

Table A2: Default hyper-parameters for SEXp, RAP, ToT. Parameters are grouped into LLM-related, general, and method-specific settings, following the settings used in SExp (Lee et al., 2025)

(a) Semantic Exploration (SExp)



(b) Latent Semantic Clustering (Ours)

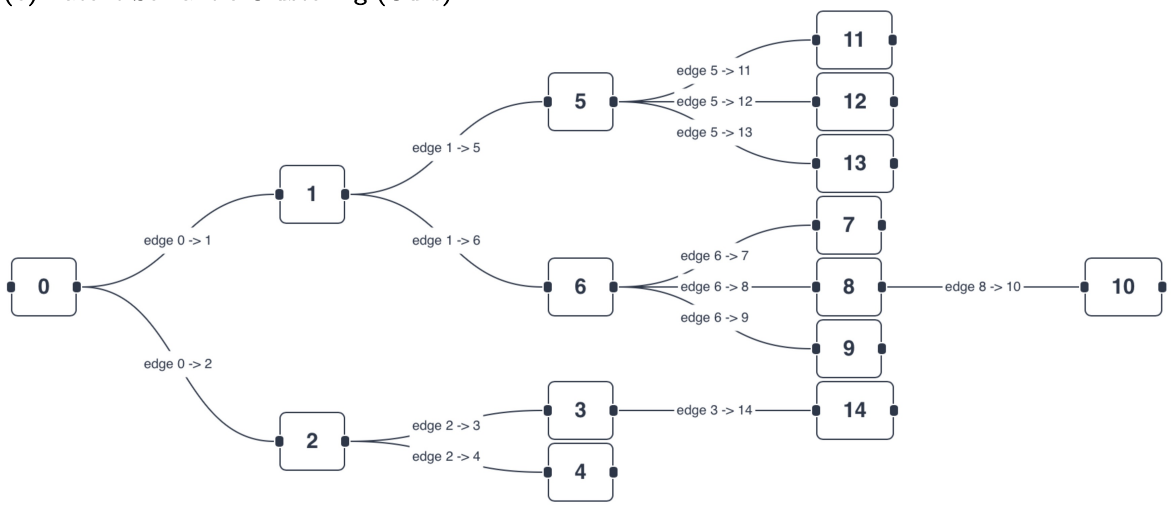


Figure A1: Reasoning trees constructed from GSM8K using Llama3-8B-Instruct, comparing SExp and SExp-LSC. SExp-LSC constructs shallower trees by leveraging context-dependent semantics, avoiding semantically redundant paths. A concrete example of generated sequences is shown in Figure 5.

Method	Module	BioASQ		SQuAD		TriviaQA	
		AUROC ↑	AUARC ↑	AUROC ↑	AUARC ↑	AUROC ↑	AUARC ↑
Llama3-8B-Instruct							
P(True) (Kadavath et al., 2022)	LLM	0.7565	0.6663	0.6744	0.3559	0.7781	**0.7926**
PE (Malinin and Gales, 2020)	-	0.8102	0.7006	<u>0.7413</u>	<u>0.3954</u>	<u>0.7839</u>	<u>0.7912</u>
KLE (Nikitin et al., 2024)	NLI	0.8361	0.6960	0.7199	0.3891	0.7771	0.7755
Deg (Lin et al., 2024)	NLI	0.7740	0.6411	0.6780	0.3466	0.6343	0.7083
EigV (Lin et al., 2024)	NLI	0.5646	0.5198	0.5877	0.2943	0.6210	0.7010
ECC (Lin et al., 2024)	NLI	0.8238	0.7060	0.6783	0.3694	0.7150	0.7677
NumSets (Kuhn et al., 2023)	NLI	0.8318	0.6794	0.7311	0.3751	0.7717	0.7700
SE (Kuhn et al., 2023)	NLI	0.8232	0.6873	0.7148	0.3876	0.7554	0.7666
DSE (Kuhn et al., 2023)	NLI	0.8338	0.6883	0.7288	0.3835	0.7783	0.7752
SE-LSC (Ours)	-	<u>0.8602</u>	**0.7222**	0.7401	0.3941	0.7745	0.7867
DSE-LSC (Ours)	-	**0.8654**	<u>0.7152</u>	**0.7616**	**0.3981**	**0.7843**	0.7855
Mistral-7B-Instruct							
P(True)	LLM	0.7217	0.5113	0.6718	0.3006	0.7905	0.6800
PE	-	0.5233	0.3831	0.7094	0.3149	0.7653	0.6555
KLE	NLI	0.8575	0.5745	0.7536	**0.3317**	<u>0.8310</u>	<u>0.7025</u>
Deg	NLI	0.7927	0.5402	0.6296	0.2675	0.6434	0.5818
EigV	NLI	0.5313	0.3704	0.4838	0.1972	0.5388	0.5096
ECC	NLI	0.7508	0.5268	0.5911	0.2628	0.6491	0.6047
NumSets	NLI	0.8553	0.5503	0.7519	0.3090	0.8164	0.6778
SE	NLI	0.8571	<u>0.5759</u>	0.7516	<u>0.3313</u>	**0.8338**	**0.7081**
DSE	NLI	0.8569	0.5587	0.7517	0.3188	0.8245	0.6927
SE-LSC (Ours)	-	<u>0.8605</u>	**0.5866**	<u>0.7538</u>	0.3284	0.8234	0.6964
DSE-LSC (Ours)	-	**0.8696**	0.5565	**0.7584**	0.3267	0.8161	0.6870

Table A3: Comparison of uncertainty quantification methods based on AUROC across three datasets using two LLMs: Llama3-8B and Mistral-7B. **Bold** and <u>underlined</u> values indicate the best and second-best performance in each setting, respectively.

An answer generation prompt for BioASQ

Input data is Answer the following question as briefly as possible.
Question: Is there a sequence bias in MNase digestion patterns?
Answer: yes
Question: Is the transcriptional regulator BACH1 an activator or a repressor?
Answer: Repressor
Question: Which fusion protein is involved in the development of Ewing sarcoma?
Answer: EWS/FLI1
Question: Is PLK2 involved in alpha-synuclein phosphorylation in the nervous system?
Answer: yes
Question: What gene is mutated in Familial Mediterranean Fever?
Answer: MEFV gene
Question: What are 'vildagliptin', 'sitagliptin', 'saxagliptin', 'alogliptin', 'linagliptin', and 'dutogliptin'?
Answer:

An answer generation prompt for SQuAD

Input data is Answer the following question as briefly as possible.
Question: Which side of the road do vehicles on Cyprus drive on?
Answer: left-hand
Question: What did John Rawls publish?
Answer: A Theory of Justice
Question: What is the attempt to understand other societies on their own terms?
Answer: cultural relativism
Question: Which park does 27th Street pass through between Ninth and Tenth Avenues?
Answer: Chelsea
Question: What caused the constant linear velocity?
Answer: Noel Pemberton Billing's patented add-on governor device
Question: Following the end of the second World War, what was a still a popular theme among films makers in Burma ?
Answer:

A reward generation prompt for TriviaQA

Input data is Answer the following question as briefly as possible.
Question: Who founded the Jaguar motor company?
Answer: william lyons
Question: The name of which Russian spacecraft means 'travelling companion' or 'satellite'?
Answer: sputnik
Question: Which record label recorded The Supremes and The Jackson 5?
Answer: motown
Question: What was the Russian City of Nizhny Novgorod called between 1932 and 1990?
Answer: gorky
Question: "Who wrote the music for the musical ""A Chorus Line""?"
Answer: marvin hamlisch
Question: What was the name of the female that politician John Profumo had an affair with which ended his political career in 1963?
Answer:

Figure A2: Example prompts for BioASQ, SQuAD, and TriviaQA. *Italic texts* denote 5-shot example.

An answer generation prompt for GSM8K

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question: ".
Question 1: Albert is wondering how much pizza he can eat in one day. He buys 2 large pizzas and 2 small pizzas. A large pizza has 16 slices and a small pizza has 8 slices. If he eats it all, how many pieces does he eat that day?
Question 1.1: How many slices are in one large pizza?
Answer 1.1: One large pizza has 16 slices. The answer is 16.
Question 1.2: How many slices are there in total from the large pizzas?
*Answer 1.2: He buys 2 large pizzas, so 2 * 16 = 32 slices. The answer is 32.*
Question 1.3: How many slices are in one small pizza?
Answer 1.3: One small pizza has 8 slices. The answer is 8.
Question 1.4: How many slices are there in total from the small pizzas?
*Answer 1.4: He buys 2 small pizzas, so 2 * 8 = 16 slices. The answer is 16.*
Question 1.5: Now we can answer the question: How many pieces does he eat that day?
Answer 1.5: There are 32 slices from the large pizzas and 16 slices from the small pizzas, so he eats 32 + 16 = 48 pieces that day. The answer is 48.
Question 2: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Question 2.1: How much did Josh spend on the house and repairs in total?
Answer 2.1:

An action generation prompt for GSM8K

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question: ".
Question 1: Albert is wondering how much pizza he can eat in one day. He buys 2 large pizzas and 2 small pizzas. A large pizza has 16 slices and a small pizza has 8 slices. If he eats it all, how many pieces does he eat that day?
Question 1.1: How many slices are in one large pizza?
Answer 1.1: One large pizza has 16 slices. The answer is 16.
Question 1.2: How many slices are there in total from the large pizzas?
*Answer 1.2: He buys 2 large pizzas, so 2 * 16 = 32 slices. The answer is 32.*
Question 1.3: How many slices are in one small pizza?
Answer 1.3: One small pizza has 8 slices. The answer is 8.
Question 1.4: How many slices are there in total from the small pizzas?
*Answer 1.4: He buys 2 small pizzas, so 2 * 8 = 16 slices. The answer is 16.*
Question 1.5: Now we can answer the question: How many pieces does he eat that day?
Answer 1.5: There are 32 slices from the large pizzas and 16 slices from the small pizzas, so he eats 32 + 16 = 48 pieces that day. The answer is 48.
Question 2: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Question 2.1:

A reward generation prompt for GSM8K

Given a question and some sub-questions, determine whether the last sub-question is useful to answer the question. Output 'Yes' or 'No', and a reason.
Question 1: Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice as 30 years old, how old is Kody?
Question 1.1: How old is Mohamed?
Question 1.2: How old was Mohamed four years ago?
New question 1.3: How old was Kody four years ago?
Is the new question useful? Yes. We need the answer to calculate how old is Kody now.
Question 2: Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
New question 2.1: How much did Josh spend on the house?
Is the new question useful?

Figure A3: Example prompts for GSM8K. *Italic texts* denote 1-shot example. The prompts are identical to those used in Lee et al. (2025).

An answer generation prompt for ARC

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question with an option from A to D: ".
Question 1: Juan and LaKeisha roll a few objects down a ramp. They want to see which object rolls the farthest. What should they do so they can repeat their investigation? Options: A) Put the objects in groups, B) Change the height of the ramp, C) Choose different objects to roll, D) Record the details of the investigation.
Question 1.1: What is necessary to ensure that experimental results can be repeated?
Answer 1.1: To ensure repeatability, experimental details must be accurately recorded. The answer is to record details.
Question 1.2: What kind of information should Juan and LaKeisha record for repeatability?
Answer 1.2: They should record details like the objects used, ramp height, and surface conditions. The answer is experimental conditions.
Question 1.3: How would recording experimental details help in the investigation?
Answer 1.3: Recording details allows them to recreate the exact same conditions for reliable comparison. The answer is that it enables consistent replication.
Question 1.4: Now we can answer the question with an option from A to D: What should they do to repeat their investigation?
Answer 1.4: Record the details of the investigation. The answer is D.
Question 2: Which method is the safest way to watch an eclipse of the Sun? Options: A) Turn away after two or three minutes. B) Look at the Sun through a long telescope. C) Cast an image through a pinhole onto a screen. D) Blink often until your eyes get used to the light..
Question 2.1: Why should you not look directly at the Sun during an eclipse?
Answer 2.1:

An action generation prompt for ARC

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question with an option from A to D: ".
Question 1: Juan and LaKeisha roll a few objects down a ramp. They want to see which object rolls the farthest. What should they do so they can repeat their investigation? Options: A) Put the objects in groups, B) Change the height of the ramp, C) Choose different objects to roll, D) Record the details of the investigation.
Question 1.1: What is necessary to ensure that experimental results can be repeated?
Answer 1.1: To ensure repeatability, experimental details must be accurately recorded. The answer is to record details.
Question 1.2: What kind of information should Juan and LaKeisha record for repeatability?
Answer 1.2: They should record details like the objects used, ramp height, and surface conditions. The answer is experimental conditions.
Question 1.3: How would recording experimental details help in the investigation?
Answer 1.3: Recording details allows them to recreate the exact same conditions for reliable comparison. The answer is that it enables consistent replication.
Question 1.4: Now we can answer the question with an option from A to D: What should they do to repeat their investigation?
Answer 1.4: Record the details of the investigation. The answer is D.
Question 2: Which method is the safest way to watch an eclipse of the Sun? Options: A) Turn away after two or three minutes. B) Look at the Sun through a long telescope. C) Cast an image through a pinhole onto a screen. D) Blink often until your eyes get used to the light..
Question 2.1:

A reward generation prompt for ARC

Given a question and some sub-questions, determine whether the last sub-question is useful to answer the question. Output 'Yes' or 'No', and a reason.
Question 1: How are particles in a block of iron affected when the block is melted? Options: A) The particles gain mass, B) The particles contain less energy, C) The particles move more rapidly, D) The particles increase in volume.
Question 1.1: What happens to particle energy when a solid melts?
Question 1.2: How does the movement of particles change during melting?
New question 1.3: Why does increased movement signify a phase change to liquid?
Is the new question useful? Yes, because understanding why increased movement signifies a phase change helps clarify the behavior of particles during melting.
Question 2: Which method is the safest way to watch an eclipse of the Sun? Options: A) Turn away after two or three minutes. B) Look at the Sun through a long telescope. C) Cast an image through a pinhole onto a screen. D) Blink often until your eyes get used to the light..
New question 2.1: Why should you not look directly at the Sun during an eclipse?
Is the new question useful?

Figure A4: Example prompts for ARC. *Italic texts* denote 1-shot example. The prompts are identical to those used in Lee et al. (2025).

A pairwise clustering label generation prompt

Given the context and a question pair, determine whether the two questions have the same meaning. Output 'Yes' or 'No', and a reason.
Context: Is a Boeing 737 cost covered by Wonder Woman (2017 film) box office receipts?
Question pair: "What is the cost of a Boeing 737?" and "How much does a Boeing 737 cost?"
Answer: Yes, because both questions ask for the cost of the same airplane.
Context: Cynthia eats one serving of ice cream every night. She buys cartons of ice cream with 15 servings of ice cream per carton at a cost of $4.00 per carton. After 60 days, how much will she spend on ice cream?
Question pair: "How many servings of ice cream does Cynthia eat in 60 days?" and "What is the total number of ice cream servings Cynthia eats?"
Answer: Yes, because both ask for the total servings of ice cream over 60 days.
Context: In a jewelers store, the price of a gold Jewell is 4/5 times as much as the price of a diamond Jewell. The cost of a silver Jewell is $400 less than the price of gold. If a diamond Jewell is $2000, find the total price for all three jewels.
Question pair: "How much is the price of a silver Jewell?" and "What is the price of the gold Jewell?"
Answer: No, because they refer to different types of Jewell.
Context: Julie is reading a 120-page book. Yesterday, she was able to read 12 pages and today, she read twice as many pages as yesterday. If she wants to read half of the remaining pages tomorrow, how many pages should she read?
Question pair: "How many pages did Julie read yesterday?" and "What is the number of pages Julie finished reading today?"
Answer:

Figure A5: An example prompt for generating pairwise clustering labels. *Italic texts* denote 3-shot example.