SENSE: Semantic Embedding Navigation with Soft-gated Evaluation for Retrieval-based Speculative Decoding

Anonymous ACL submission

Abstract

Speculative Decoding (SD) accelerates Large Language Model (LLM) inference by employing a lightweight draft model to propose candidate tokens, which are verified in parallel by the target model, without compromising generation quality. While Retrieval-based Speculative Decoding (RSD) is favored for its plug-and-play versatility, its potential is impeded by rigid lexical dependencies, rendering both retrieval and verification brittle to surface-level variations. To address this, we propose ***SENSE*** (**S**emantic **E**mbedding **N**avigation with **S**oft-gated **E**valuation). By anchoring retrieval on the hidden states of the target model, *SENSE* establishes robust semantic alignment, which enables Soft-gated Evaluation to relax strict surface-form matching and retain target-supported lexical alternatives under controlled verification. Extensive experiments across diverse domains demonstrate that *SENSE* outperforms multiple baselines on the LLaMA and Qwen families, attaining up to **4.09** mean acceptance length and **3.26**× speedup while maintaining comparable generation quality under the evaluated settings. We will release our codes.

1 Introduction

While large language models (LLMs) leverage parameter scaling to maximize expressivity (Kaplan et al., 2020; Chowdhery et al., 2022), their inference efficiency suffers from the sequential nature of auto-regressive decoding (Zeng et al., 2023; Guo et al., 2025). Among diverse optimization paradigms (Zhou et al., 2024), Speculative Decoding (SD) distinguishes itself through its structural decoupling. By leveraging a lightweight draft model, SD accelerates generation while remaining inherently agnostic to the target model's architecture (Leviathan et al., 2023).

Within the SD landscape, Retrieval-based Speculative Decoding (RSD) stands out for its plug-and-play universality, effective across heterogeneous model architectures and vocabularies without modification (Somasundaram et al., 2024). However, their flexibility is inherently constrained by the rigidity of exact-match rule. Consequently, many semantically aligned segments are erroneously rejected due to discrete token-level discrepancies. Inspired by lossely SD strategies (Garipov et al., 2025) that prioritize semantic equivalence over lexical exactness, we posit a critical question: Can the synergy of retrieval-augmented drafting and semantic-aware verification liberate RSD from the limitations of strict lexical matching?

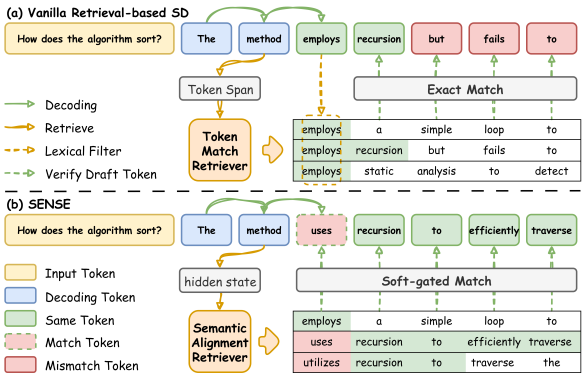

Figure 1: (a) Vanilla RSD retrieves tokens using an exact match rule based on token spans. This approach suffers from lexical limitations, as it strictly requires the retrieved drafts to exactly match the next token "employs". (b) *SENSE* utilizes hidden states to perform semantic alignment retrieval. This allows it to retrieve semantically similar drafts and verify them through a soft-gated match mechanism, effectively overcoming rigid lexical constraints.

A critical bottleneck persists in the retrieval phase: the rigidity of lexical-based retrieval keys. Mainstream paradigms typically anchor retrieval solely to the target model's immediate token prediction, imposing strict prefix constraints that severely circumscribe the candidate pool (He et al., 2024; Ho et al., 2024). While recent methodologies like

AASD (Wang et al., 2025) and SpecLogic (Liu et al., 2026) attempt to alleviate this limitation by expanding the search space, they remain confined by the inherent rigidity of token-level exact matching. Consequently, these strategies fail to capture the rich semantic nuances inherent in generation, rendering the retrieval mechanism blind to candidates that are semantically equivalent yet lexically distinct.

Furthermore, the intrinsic nature of retrieved candidates fundamentally constrains verification flexibility. The absence of generative metadata such as specifically probabilistic logits, these candidates are incompatible with advanced distribution alignment heuristics (Li et al., 2026). Moreover, while retrieval scales to massive candidate pools, the resulting large batch sizes impose severe computational bottlenecks on conventional linear verification strategies (Li et al., 2025a), thereby negating potential efficiency gains.

In this paper, we introduce ***SENSE*** (**S**emantic **E**mbedding **N**avigation with **S**oft-gated **E**valuation) to overcome these problems. By anchoring on semantic alignment, *SENSE* synergizes an embedding-based retrieval with a complementary soft-gated evaluation. To overcome lexical rigidity, we introduce **Semantic Embedding Navigation (SEN)**, which anchors retrieval on the target model's hidden states rather than discrete token predictions. By querying both static datastores and dynamic contexts, *SEN* retrieves candidates aligned with the model's generative intent, even when their surface forms diverge from the greedy prediction. Complementing the *SEN*, we introduce **Soft-gated Evaluation (SE)**, an adaptive verification protocol designed to retain target-supported candidates beyond strict lexical matching. Driven by entropy-guided logic, *SE* enforces strict exact matching in high-confidence regions while adaptively validating high-uncertainty spans through distributional top-k alignment and parallelized neighborhood fusion to minimize verification overhead.

To summarize, our contributions are two-fold. First, we propose ***SENSE*** (**S**emantic **E**mbedding **N**avigation with **S**oft-gated **E**valuation), a training-free retrieval-verification framework that improves candidate utilization in RSD. By retrieving continuations from target-model hidden states and applying entropy-guided relaxed verification, *SENSE* alleviates the lexical rigidity of conventional retrieval and exact matching. Second, we conduct extensive experiments on LLaMA and Qwen backbones, where *SENSE* achieves up to **3.26**$\times$ speedup and a mean acceptance length of **4.09** tokens per decoding step while maintaining comparable downstream generation quality. Further analyses reveal the drafting and verification advantages that drive *SENSE*'s efficiency.

2 Related Work

Retrieval-based speculative decoding accelerates LLM inference through a *retrieve-verify* paradigm: a lightweight retriever fetches candidates from a datastore, followed by parallel verification using the target model (Hu et al., 2025; Ryu and Kim, 2024). We structure the related literature along two dimensions: retrieval-based drafting and relaxed verification strategies.

2.1 Retrieval-Based Drafting

Methods of Retrieval-Based Drafting (Yang et al., 2023; He et al., 2024; Hu et al., 2024; Ho et al., 2024) fetch candidate continuations from external datastores. Sparse approaches rely on exact n-gram matching, rendering them susceptible to surface-form brittleness; conversely, dense retrieval (Gritta et al., 2025) mitigates this limitation by leveraging semantic embeddings. Similarly, hybrid strategies (Somasundaram et al., 2024; Liu et al., 2026; Stewart et al., 2024) and heterogeneous architectures (Divilkovskiy et al., 2025) further expand the search scope. However, despite the expanded retrieval space offered by these methods, they remain constrained by *lexical rigidity* during the subsequent filtering and verification phases (Wang et al., 2025). We address this limitation by anchoring retrieval on hidden states and integrating distributional constraints to relax these rigid boundaries.

2.2 Relaxed Verification Strategies

Exact-match verification rigidly rejects semantically valid but non-identical drafts (Holtzman et al., 2020). Training-based relaxations (Bachmann et al., 2025; Garipov et al., 2025) circumvent strict token matching by learning semantic judgments, whereas training-free approaches (Li et al., 2025a; Wang et al., 2025; Li et al., 2026) exploit entropy metrics or risk bounds. Crucially, these methods are constrained by unidimensional criteria along linear paths, causing a cascading loss of valid future tokens upon early failure. We synergize top-k membership and neighborhood-fused error density to robustly verify parallel trajectories.

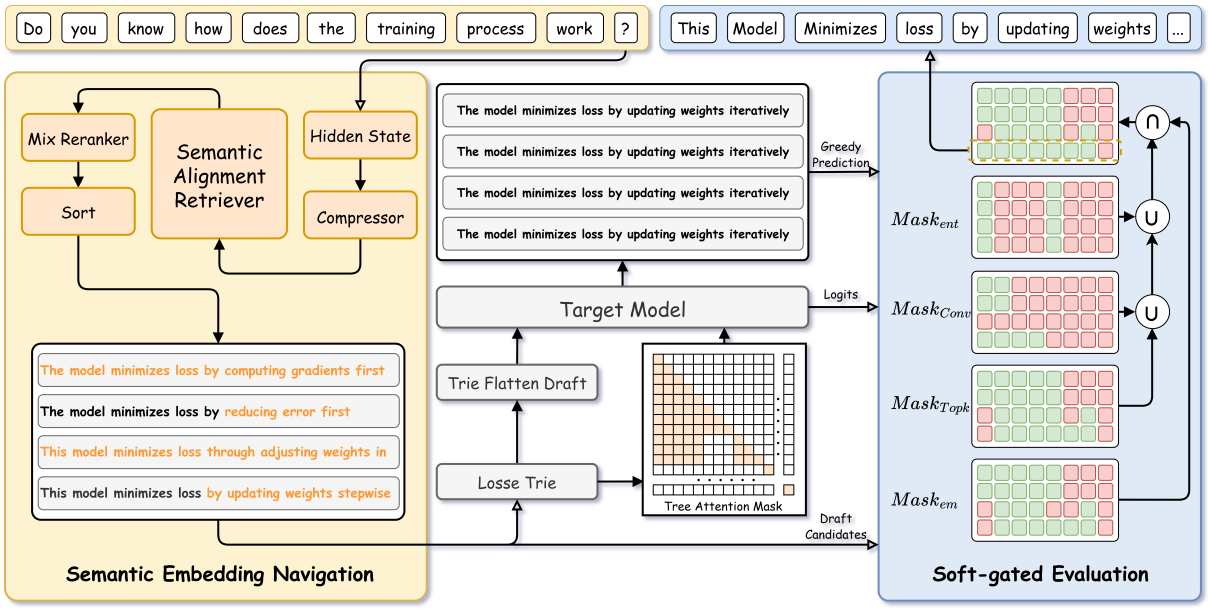

Figure 2: Overview of the SENSE framework. (Left) The Semantic Embedding Navigation module retrieves semantically aligned drafts based on the input's hidden states. These drafts are flattened into a Tree-Structured Parallelism for efficient verification by the target model. (Right) The Soft-gated Evaluation module filters the verification results using a cascade of masks to robustly determine the final accepted tokens.

3 Method

Following the preliminaries (Section 3.1), we introduce *SENSE* (Figure 2), a training-free framework for controlled lossy retrieval-based speculative decoding. SENSE combines Semantic Embedding Navigation (SEN) (Section 3.2) to retrieve candidate continuations from target-model hidden states with Soft-gated Evaluation (SE) (Section 3.3) to accept target-supported alternatives that strict token matching would discard. Through this co-design, *SENSE* improves candidate utilization while keeping the target model as the verifier.

3.1 Preliminaries

Given the iterative nature of SD, we focus on the operational logic for generating a specific token x_t. Let $q_{<t} = [x_1, x_2, \ldots, x_{t-1}]$ denote the prefix sequence. To generate x_t, we process $q_{<t}$ through the target model LLM to obtain the hidden state and target next-token distribution, denoted as $h_{q_{<t}}, p_{q_{<t}} = LLM(q_{<t})$. From these outputs, we derive the greedy anchor $\tilde{x}_t = \text{argmax}_{v \in \mathcal{V}} p_{q_{<t}}[v]$ and the projected query $k_t = F(h_{q_{<t}}, \tilde{x}_t)$ for retrieving semantically aligned candidates.

Formally, the corpus $C_{db} = \{(k_i, c_i) \mid k_i \in \mathbb{R}^h, c_i \in V^*\}$ establishes a mapping from the query embedding space $\mathbb{R}^h$ to text sequences. Building upon this representation, the retriever R identifies the N most relevant sequences (each with maximum length M) from C_{db} according to a similarity metric, constructing the candidate draft set C_{cand}:

$$C_{cand} = R(k_t, C_{db}) = \{c^1, c^2, \ldots, c^N\} \quad (1)$$

To verify C_{cand}, each draft is concatenated with $q_{<t}$ and run a batched forward pass through the target verifier, denoted as $\mathcal{O} = LLM_{verif}(q_{<t}, C_{cand})$.

For each draft $c^n = [x'_1, \ldots, x'_M]$, the batched verifier returns a sequence of one-step-ahead distributions $P^n = [p^n_0, \ldots, p^n_M]$. For $j = 0, \ldots, M-1$, p^n_j predicts the token at position $t + j$ conditioned on $q_{<t}$ and the consumed draft prefix $[x'_1, \ldots, x'_j]$, and is therefore aligned with the candidate token x'_{j+1}. The final distribution p^n_M is a lookahead output used to continue decoding after the full draft is accepted. Accordingly, the strict verification baseline over draft positions is $\hat{y}^n = [\text{argmax}(p^n_0), \ldots, \text{argmax}(p^n_{M-1})]$.

Finally, the verifier function V determines the acceptance length ℓ^n by evaluating the draft c^n against the predicted distributions P^n and the baseline $\hat{y}^n$. The optimal accepted path d^* is selected to maximize this length:

$$\begin{aligned} \ell^n &= V(c^n, \hat{y}^n, \mathcal{O}) \\ d^* &= \underset{c^n \in C_{cand}}{\text{argmax}}\, \ell^n, \quad \text{where } 0 \le \ell^n \le M \end{aligned} \quad (2)$$

The upper bound M corresponds to accepting all draft tokens. If $\ell^n = 0$, the draft is rejected entirely and decoding falls back to the anchor token at position t.

3.2 Semantic Embedding Navigation

Hybrid semantic datastore. Semantic Embedding Navigation (SEN) uses target-model hidden states as retrieval keys, so that candidate drafts are selected by representation-level proximity rather than surface overlap alone. We maintain a hybrid datastore that combines a static datastore C_s with a dynamic context buffer C_d. The static datastore provides reusable suffixes collected offline, while the dynamic buffer records continuations generated within the current decoding instance. Each entry stores a compressed hidden-state key and its following token sequence:

$$\begin{aligned} C_{db} &= C_s \cup C_d \\ &= \{(F(h_i), s_i) \mid h_i \in \mathbb{R}^d,\ s_i \in \mathcal{V}^*\}, \end{aligned} \tag{3}$$

where $F : \mathbb{R}^d \to \mathbb{R}^h (h \ll d)$ denotes a projection operator for reducing the storage cost of high-dimensional hidden states (Gritta et al., 2025). The datastore serves only as a source of speculative continuations, and its contents are never incorporated into the target model's conditioning context.

Retrieval and candidate organization. At decoding step t, SEN queries C_{db} with the projected target state $k_t = F(h_{q<t})$ and retrieves an initial pool of N' nearest suffixes by ANN search (Douze et al., 2024):

$$D_{ret} = \mathrm{ANN}(k_t, C_{db}, N') = \{(k_i, s_i)\}_{i=1}^{N'}. \tag{4}$$

Unlike exact-match retrieval, this step does not require the first token of a retrieved suffix to match the greedy anchor $\tilde{x}_t$. We then rerank the retrieved pool with a composite score that prioritizes exact first-token matches while retaining representation-aligned alternatives:

$$\begin{aligned} S(s_i) &= \lambda_{\text{lex}} \mathbb{I}(s_i[1] = \tilde{x}_t) \\ &\quad + \lambda_{\text{sem}} \operatorname{sim}(k_t, k_i), \\ D_{cand} &= \underset{(k_i,s_i)\in D_{ret}}{\text{Top-N}} \quad S(s_i), \end{aligned} \tag{5}$$

where $\lambda_{\text{lex}} \gg \lambda_{\text{sem}}$.This reranking keeps exact first-token matches as preferred candidates, while semantic similarity orders or supplements the remaining proposals.

Before verification, we organize D_{cand} into a compact prefix-sharing structure based on Tree-Structured Parallelism (Miao et al., 2024). Specifically, we sort candidate suffixes, merge adjacent common prefixes, and keep an inverse mapping Γ from flattened positions back to candidate paths. This produces a flattened input $\mathcal{T}_{flat}$ and a topology-aware causal mask A_{tree} for a single batched verifier pass. The verifier outputs flattened hidden states and next-token distributions, $(H_{\text{flat}}, P_{\text{flat}}) = LLM_{verif}(q_{<t}, \mathcal{T}_{flat}; A_{\text{tree}})$, which are restored by Γ to the original candidate layout:

$$\begin{aligned} H^{n,0}_{\text{align}} &= h_{q<t}, \quad H^{n,j}_{\text{align}} = \Gamma(H_{\text{flat}})^{n,j}, \\ P^{n,0}_{\text{align}} &= p_{q<t}, \quad P^{n,j}_{\text{align}} = \Gamma(P_{\text{flat}})^{n,j}, \quad j \geq 1. \end{aligned} \tag{6}$$

Here $P^{n,j}_{\text{align}}$ corresponds to the distribution p^n_j defined in Section 3.1, while $H^{n,j}_{\text{align}}$ provides the corresponding hidden state used to continue decoding after a prefix is accepted.

3.3 Soft-gated Evaluation

Given the candidate paths organized by SEN, Soft-gated Evaluation (SE) determines how many tokens can be accepted from each retrieved draft. Let $X \in \mathcal{V}^{N\times M}$ denote the aligned candidate-token matrix, where $X_{n,j}$ is the token proposed by draft n at position j for $j = 0, \ldots, M-1$, and $P^{n,j}_{\text{align}}$ denotes its verifier distribution. The lookahead distribution $P^{n,M}_{\text{align}}$ is retained only for continuing decoding after prefix acceptance and is excluded from the acceptance masks below. Since SEN may retrieve candidates that differ from the greedy trajectory in surface form, strict exact matching would discard many representation-aligned continuations. SE therefore keeps exact matching as the default path, while allowing mismatches only when the target model is uncertain and the candidate satisfies target-side support checks.

Exact path and uncertainty gate. We first derive the greedy verifier prediction and its exact-match mask for each candidate position:

$$\begin{aligned} Y_{n,j} &= \underset{v\in\mathcal{V}}{\operatorname{argmax}}\, P^{n,j}_{\text{align}}(v), \\ B^{n,j}_{\text{em}} &= \mathbb{I}(X_{n,j} = Y_{n,j}). \end{aligned} \tag{7}$$

Exact matches are accepted directly, whereas mismatches enter relaxed verification only at uncertain positions. Following prior work that uses entropy to characterize model uncertainty (Wang et al., 2025; Bachmann et al., 2025), we define:

$$\begin{aligned} E_{n,j} &= -\sum_{v\in\mathcal{V}} P^{n,j}_{\text{align}}(v) \log P^{n,j}_{\text{align}}(v), \\ B^{n,j}_{\text{ent}} &= \mathbb{I}(E_{n,j} > \gamma). \end{aligned} \tag{8}$$

When $B_{\text{ent}}[n, j] = 0$, a non-exact token is rejected and SE reduces to strict verification at that position.

Target-supported relaxed acceptance. For positions where the entropy gate opens, SE applies two target-side support checks. The first is a Top-k criterion (Holtzman et al., 2020; Leviathan et al., 2023), which allows a candidate token only if it lies in a high-probability region of the target distribution:

$$\begin{aligned} \mathcal{Y}^{n,j}_{\text{topk}} &= \text{TopK}_k\left(P^{n,j}_{\text{align}}\right), \\ B^{n,j}_{\text{topk}} &= \mathbb{I}\left(X_{n,j} \in \mathcal{Y}^{n,j}_{\text{topk}}\right). \end{aligned} \tag{9}$$

This criterion prevents retrieved tokens with low target-model support from being accepted through the relaxed path.

The second check is local consistency. Inspired by sequential lookahead verification (Li et al., 2025a), we use **Parallelized Neighborhood Fusion** to measure whether a mismatch is isolated or part of a divergent region. Given the mismatch mask $\bar{B}_{\text{em}} = \mathbf{1} - B_{\text{em}}$, we compute its local error density with a length-W unit kernel:

$$\begin{aligned} \mathbf{R} = (\bar{B}_{\text{em}} \oplus \mathbf{1}_{pad}) * \mathbf{k}, \quad \mathbf{k} = [1]^W, \\ B_{\text{con}} = \mathbb{I}(\mathbf{R} \leq \rho). \end{aligned} \tag{10}$$

Here ρ controls the tolerated local error density; together, B_{topk} and B_{con} constrain relaxed acceptance by target probability support and local consistency.

Final Decision. We combine the exact path, uncertainty gate, and two relaxed support checks into the final acceptance mask. A token is accepted either by exact matching, or by the relaxed path when the target model is uncertain and at least one support check is satisfied:

$$G = B_{\text{em}} \vee (B_{ent} \wedge (B_{topk} \vee B_{con})) \tag{11}$$

The final mask G induces an accepted prefix length L_n for each draft, defined by the first position that violates the acceptance criterion. We then select the draft with the longest accepted prefix:

$$\begin{aligned} L_n &= \min\left(\{j \mid G^{n,j} = 0\} \cup \{M\}\right), \\ n^* &= \underset{n \in \{1,\ldots,N\}}{\text{argmax}}\ L_n. \end{aligned} \tag{12}$$

Following verification, we update the decoding state by indexing into the aligned verifier outputs. Since L_{n^*} denotes the accepted prefix length, the continuation state for the next iteration is given by the aligned hidden state and logits at position L_{n^*} of the selected draft n^*:

$$h_{\text{next}} = H^{n^*,L_{n^*}}_{\text{align}}, p_{\text{next}} = P^{n^*,L_{n^*}}_{\text{align}} \tag{13}$$

4 Experiments

4.1 Experimental Setup

Benchmarks and Datastores. We evaluate *SENSE* on four primary domains: mathematics (GSM8K (Cobbe et al., 2021)), coding (CodeAlpaca (Chaudhary, 2026)), dialog (UltraChat (Ding et al., 2023)), and question answering (TriviaQA (Joshi et al., 2017)). We further use LongBench-v2 (Bai et al., 2025) for a long-context stress test. Critically, we construct both In-Distribution (ID) and Out-Of-Distribution (OOD) datastores: ID datastores contain responses generated by the target LLM itself, while OOD datastores use original ground-truth labels. This distinction enables rigorous assessment of datastore-model alignment effects. Details are in Section E.1.

Models. We evaluate on six target models across two spanning families: Llama-2 (7B/13B) (Touvron et al., 2023) and Qwen (2.5-7B/14B-Instruct (Team, 2024), 3-8B/14B (Team, 2025)). For generative baselines, we pair these with corresponding draft models (MicroLlama (Wang, 2024), Qwen2.5-0.5B-Instruct, Qwen3-0.6B).

Baselines. We compare against retrieval-based methods (REST (He et al., 2024), DReSD (Gritta et al., 2025), PLD (Somasundaram et al., 2024)), generative drafting (SpD (Leviathan et al., 2023), EAGLE-3 (Li et al., 2025b)), and relaxed verification approaches (ARC (Li et al., 2026), FLY (Li et al., 2025a)). All baselines use their reported optimal configurations, with details in Section E.3.

Metrics. We report end-to-end speedup over vanilla target-model decoding and mean accepted tokens per decoding step (τ) to evaluate inference efficiency. Task-specific metrics for generation quality: accuracy for GSM8K and TriviaQA and ROUGE-L for CodeAlpaca and UltraChat.

Implementation. Hidden states are projected via PCA and indexed using FAISS (Douze et al., 2024) with IVF-PQ. At inference, we retrieve N=3 candidates with maximum draft length M=10. Verification uses entropy threshold θ_e=0.05 and mismatch window w=6, corresponding to γ and W in Section 3.3. All methods use greedy decoding with temperature= 0 and FP32 precision. For fair comparison, we implement all methods in a unified drafting–connector–verification framework. Framework details and full hyperparameters are provided in Sections E.4 and E.5.

Model	Method	GSM8K		CodeAlpaca		UltraChat		TriviaQA		Mean	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Llama2-7B	SpS	1.94	2.73	1.50	2.15	1.47	2.15	1.23	1.65	1.53	2.17
	PLD	4.18	3.64	2.73	2.13	2.91	2.46	3.18	6.12	3.25	3.59
	REST	2.00	1.43	2.58	1.88	2.27	1.88	2.00	1.83	2.21	1.76
	DReSD	1.37	1.38	1.87	2.47	1.07	1.58	1.25	1.64	1.39	1.77
	SENSE(OOD)	**6.44**	**6.69**	**7.88**	**8.36**	**4.77**	**5.50**	**8.23**	**8.61**	**6.83**	**7.29**
	SENSE(ID)	5.45	5.80	6.52	6.87	3.28	3.44	7.46	7.78	5.68	5.98
Llama2-13B	SpS	0.73	1.84	1.42	2.25	1.31	2.00	1.57	2.44	1.26	2.13
	PLD	1.07	**7.37**	1.54	2.79	1.54	1.95	1.70	4.75	1.46	4.22
	REST	0.44	1.04	1.97	2.25	1.35	1.64	0.54	1.03	1.08	1.49
	DReSD	0.61	1.60	0.85	2.28	1.04	2.81	1.22	3.27	0.93	2.49
	SENSE(OOD)	1.12	1.20	1.95	2.08	1.64	2.84	1.09	1.16	1.46	1.82
	SENSE(ID)	**1.24**	2.44	**3.72**	**6.83**	**1.45**	**2.73**	**5.36**	**9.75**	**2.95**	**5.44**
Qwen2.5-7B	SpS	0.78	2.34	0.81	2.36	0.61	1.88	0.67	2.00	0.71	2.14
	PLD	2.16	2.03	1.64	1.50	1.42	1.49	1.64	1.44	1.71	1.61
	REST	1.44	1.26	2.69	2.48	1.63	1.44	1.38	1.19	1.78	1.59
	DReSD	1.21	1.41	2.41	1.72	1.38	1.59	1.09	1.20	1.52	1.48
	SENSE(OOD)	1.51	1.91	2.34	3.11	1.48	2.18	1.14	1.42	1.62	2.16
	SENSE(ID)	**2.94**	**3.44**	**3.30**	**3.95**	**2.59**	**3.04**	**2.72**	**3.63**	**2.89**	**3.52**
Qwen2.5-14B	SpS	1.09	2.26	1.44	2.31	0.88	1.82	1.01	2.08	1.10	2.11
	PLD	2.12	2.02	1.63	1.41	1.45	1.38	1.68	0.83	1.72	1.41
	REST	1.38	1.23	**2.54**	2.33	1.64	1.49	1.36	1.20	1.73	1.56
	DReSD	1.33	1.29	2.26	2.39	1.82	1.75	1.46	1.50	1.71	1.73
	SENSE(OOD)	1.61	1.97	2.15	2.69	1.62	1.74	1.25	1.53	1.66	1.98
	SENSE(ID)	**3.00**	**3.48**	2.48	**2.90**	**2.40**	**2.98**	**2.47**	**2.56**	**2.59**	**2.96**
Qwen3-8B	SpS	1.34	1.98	1.61	2.02	0.68	1.70	0.72	1.75	1.09	1.86
	PLD	1.41	1.81	1.30	1.35	1.26	1.37	1.25	1.38	1.30	1.48
	REST	1.52	1.31	1.71	1.47	1.51	1.31	1.38	1.16	1.53	1.31
	DReSD	1.05	1.32	1.51	2.31	1.09	1.38	0.84	1.22	1.12	1.56
	SENSE(OOD)	1.63	1.73	1.86	2.34	1.62	1.75	1.12	1.36	1.56	1.80
	SENSE(ID)	**2.48**	**2.63**	**2.75**	**3.02**	**2.09**	**2.18**	**2.22**	**2.13**	**2.39**	**2.49**
Qwen3-14B	SpS	0.58	2.27	1.05	1.99	0.60	1.79	0.52	1.92	0.69	1.99
	PLD	0.91	1.71	1.53	1.39	1.05	1.31	0.82	1.37	1.08	1.44
	REST	0.86	1.31	1.09	1.14	1.23	1.28	0.60	1.16	0.95	1.22
	DReSD	0.69	1.32	1.20	1.41	1.03	1.38	0.68	1.21	0.90	1.33
	SENSE(OOD)	0.98	1.67	1.21	1.63	1.83	1.59	1.37	1.31	1.35	1.55
	SENSE(ID)	**2.73**	**3.35**	**3.23**	**4.03**	**3.26**	**6.57**	**2.89**	**2.55**	**3.03**	**4.13**

Table 1: Training-free method comparison reporting Speedup Ratio and Mean Accepted Tokens (τ). Vanilla LLM serves as the 1.0 baseline. **Bold** and underlined text denote the best and second-best results, respectively.

4.2 Performance Comparison

Comparison with Training-Free Baselines. Table 1 shows *SENSE(ID)* typically dominates by minimizing distributional divergence, with mean speedups of **2.39–3.03**$\times$. *SENSE(OOD)* maintains high stability over other RSD methods. However, Llama2-7B presents a boundary condition where OOD prevails. We attribute this to the entropy-alignment trade-off: unlike confident strong models, weak models exhibit high intrinsic uncertainty, allowing OOD datastores to function as semantic correctives. Crucially, our entropy-based gating (B_{ent}) exploits this uncertainty to accept these corrective drafts, transforming generation noise into acceleration opportunities (Detailed boundary analysis in Appendix Section D.1).

Comparison with Training-Based Drafting. As detailed in Table 2, *SENSE* effectively challenges the dominance of training-based paradigms. Despite a datastore construction time significantly lower than EAGLE's training overhead, *SENSE* surpasses the SOTA method EAGLE series across most metrics. Specifically, on Qwen3-14B, *SENSE* achieves superior mean speedups of **2.98**$\times$. Crucially, the consistently higher mean accepted tokens (τ) validate that our retrieval-verification synergy generates higher-quality drafts than trained speculative heads, effectively bridging the efficiency gap between training-free and training-based approaches.

Model	Method	GSM8K		CodeAlpaca		Mean	
		Speedup	τ	Speedup	τ	Speedup	τ
Qwen2.5-14B	EAGLE-2	2.19	2.43	1.49	**2.94**	1.84	2.69
	SENSE(ID)	**3.00**	**3.48**	**2.48**	2.90	**2.74**	**3.19**
Qwen3-8B	EAGLE-3	1.71	2.00	1.77	2.02	1.74	2.01
	SENSE(ID)	**2.48**	**2.63**	**2.75**	**3.02**	**2.61**	**2.83**
Qwen3-14B	EAGLE-3	1.65	1.87	1.51	1.78	1.58	1.83
	SENSE(ID)	**2.73**	**3.35**	**3.23**	**4.03**	**2.98**	**3.69**

Table 2: Comparison with training-based methods. **Bold** indicates the best performance in each column.

Comparison with Relaxed Verification. To isolate verification effects, we benchmark SE against relaxed verification baselines ARC (Li et al., 2026) and FLY (Li et al., 2025a), while keeping the retrieval module (*SEN*) fixed. As shown in Figure 3, *SE* yields **16–22**% speedup gains on Qwen and Llama2-13B by retaining target-supported alternatives via synergistic masking that risk-bounded heuristics prune. The marginal exception on Llama2-7B aligns with the calibration constraints of weaker models discussed above, reinforcing that entropy-guided verification requires reliable probability estimates. Detailed analysis in Section D.2.

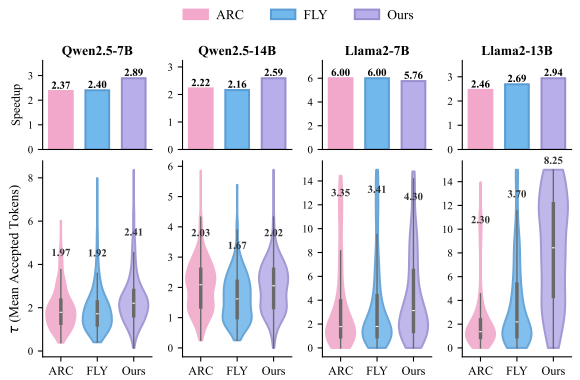

Figure 3: Verification method comparison. Upper is speedup and lower is τ.

Generation Quality. For *SE* allows controlled relaxed acceptance, we compare the downstream task metrics of *SENSE* against vanilla target-model decoding. As shown in Table 3, *SENSE* retains an average metric preservation ratio of **98.02%** across various models and tasks, indicating comparable generation quality. Most configurations remain close to the vanilla baseline, while several CodeAlpaca and TriviaQA settings slightly exceed it, suggesting that relaxed acceptance does not induce systematic quality degradation in our evaluation. Absolute task metrics are reported in Section D.4.

Model / Datastore		GSM8K	TriviaQA	CodeAlpaca	UltraChat	Avg
		ACC	ACC	ROUGE-L	ROUGE-L	
Qwen3-8B	OOD	97.8	95.6	103.7	91.7	97.2
	ID	96.8	93.3	101.9	91.3	95.8
Qwen3-14B	OOD	100.0	100.0	98.0	100.5	99.6
	ID	101.1	100.0	103.1	96.1	100.1
Qwen2.5-7B	OOD	100.0	97.9	98.3	90.5	96.7
	ID	97.8	91.7	101.0	96.4	96.7
Qwen2.5-14B	OOD	100.0	100.0	102.2	100.0	100.5
	ID	93.7	103.0	102.2	91.2	97.5
Overall mean						**98.02**

Table 3: Generation quality of *SENSE*, reported as task-metric preservation ratios relative to vanilla decoding (%). Absolute metrics are provided in Section D.4.

Long-Context Stress Test. Table 4 extends evaluation to LongBench-v2 (Bai et al., 2025) at 8k/16k; *SENSE*(ID) retains 2.36×/2.34× speedup with *comparable* accuracy to vanilla (0.25/0.36), outperforming PLD (Somasundaram et al., 2024) and DReSD (Gritta et al., 2025) by a wide margin and showing that semantic retrieval scales beyond training-length regimes.

Method	8k			16k		
	Acc	τ	Spd	Acc	τ	Spd
Vanilla	0.25	-	-	0.36	-	-
PLD	0.25	1.73	1.31×	0.34	1.86	1.85×
DReSD	0.25	1.37	1.38×	0.36	1.45	1.90×
SENSE (OOD)	0.23	1.89	1.50×	0.36	1.60	1.92×
SENSE (ID)	**0.25**	**3.10**	**2.36×**	**0.37**	**2.29**	**2.34×**

Table 4: Long-context stress test on LongBench-v2 with Qwen3-8B at 8k and 16k input lengths.

4.3 Details Analysis

Hyperparameter Sensitivity. As shown in Figure 4, we conduct a grid search on θ_e, Top-k, and window size w, revealing a clear control hierarchy: **(1) Primary Gate (θ_e):** High values (0.30) block relaxed verification, collapsing all configurations to the baseline (Acc=0.93, Spd=2.2×). **(2) Trade-off Regulator (k):** At moderate θ_e (0.01–0.10), Top-k dictates the balance. Large k (10) yields extreme speed (8.2×) but poor accuracy (16%), while small k ensures semantic alignment. **(3) Negligible Factor (w):** kernel size w has minimal impact and is aggregated in the heatmap. Our default configuration (θ_e=0.05, k=3) achieves the optimal accuracy-speedup balance.

Ablation Study. We conduct a component-wise analysis to isolate the distinct contribution of each module within *SENSE*. As shown in Table 5, ablating *SEN*(*SE*) leads to average speedup reductions

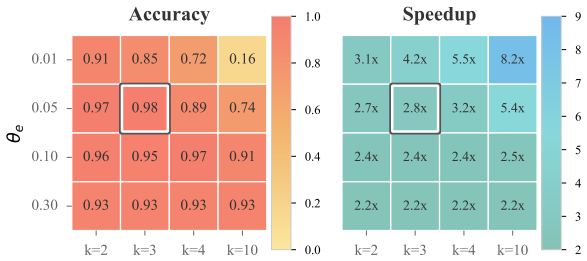

Figure 4: Hyperparameter sensitivity on Qwen2.5-7B (GSM8K). Left: accuracy; right: speedup. Each cell averages over window sizes $w \in \{5, 6, 7, 8, 9\}$; the highlighted cell marks the default setting (θ_e=0.05, k=3), achieving 98% accuracy and 2.8× speedup.

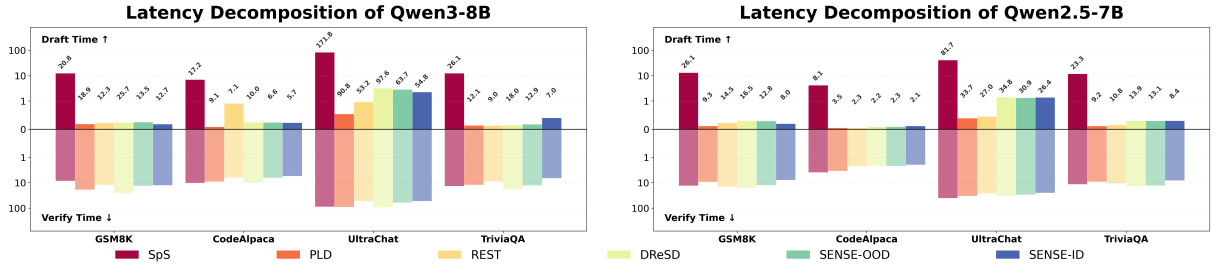

Figure 5: Latency decomposition per decoding step: drafting (upper) and verification (lower). Top annotations show total latency; **a symmetric log scale handles the magnitude gap.**

of **0.29**(**0.20**) on ID and **0.35**(**0.32**) on OOD respectively. Specifically on GSM8K, ablating *SEN* (*SE*) causes speedups to drop to **2.07**× (**2.21**×) on ID and **1.14**× (**1.04**×) on OOD. This significant decay validates our design, where *SEN* and *SE* are mutually reinforcing under the framework of semantic alignment, balancing candidate diversity with verification flexibility.

Datastore	Method	GSM8K		CodeAlpaca		UltraChat		TriviaQA	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
OOD	*SENSE*	**1.58**	**1.70**	**2.44**	**3.13**	**1.87**	**2.19**	**1.39**	**1.36**
	w/o SEN	1.14	1.35	2.16	2.34	1.36	1.41	1.24	1.24
	w/o SE	1.05	1.31	2.28	2.41	1.56	1.64	1.12	1.35
ID	*SENSE*	**2.48**	**2.64**	**2.44**	**3.13**	**2.07**	**1.18**	**2.22**	**2.12**
	w/o SEN	2.07	2.20	2.16	2.34	1.86	1.70	1.97	1.84
	w/o SE	2.21	2.49	2.28	2.41	1.84	1.91	2.07	1.98

Table 5: Component-wise ablation study of *SENSE* using Qwen3-8B across four datasets. We evaluate the impact of the SEN and SE modules on Speedup and Mean Accepted Tokens (τ).

Datastore Analysis. We stress-test the datastore along two axes in Table 6. **(1) Scale & domain alignment:** panel (a) traces a monotonic retrieval-quality spectrum, as ID coverage shrinks from 100% to 25%, speedup degrades smoothly (2.75× → 2.26×), and further drops under cross-domain (UltraChat-ID, 1.94×) and OOD (1.86×). This confirms that both *scale* and *domain alignment* contribute independently. **(2) Construction cost:** panel (b) reports a one-shot preprocessing budget on Longbench-v2 (Bai et al., 2025); the resulting datastore is amortized across all inference queries. Full size-scaling analyses and cross-domain matrices are deferred to Sections D.10 and D.11.

Latency Analysis. We decompose online latency into drafting and verification in Figure 5. In the drafting stage, *SENSE* avoids the repeated forward passes required by generative draft models and instead retrieves candidate continuations with substantially lower overhead. This advantage is more pronounced in longer contexts, where draft models process growing prefixes while retrieval remains

(a) Retrieval-quality spectrum on CodeAlpaca.

Datastore configuration	Speedup	τ
CodeAlpaca-ID (100%)	**2.75**	**3.02**
CodeAlpaca-ID (75%)	2.47	2.76
CodeAlpaca-ID (50%)	2.45	2.76
CodeAlpaca-ID (25%)	2.26	2.47
CodeAlpaca-OOD	1.86	2.34
UltraChat-ID (cross-domain)	1.94	2.31

(b) Offline construction cost (LongBench-v2, 465 MB).

RAW Emb	Train PCA	Final Emb	Index	Total
21m49s	8m46s	28m23s	3m24s	62m22s

Table 6: Datastore analysis of *SENSE* with Qwen3-8B: (a) retrieval quality under ID scaling, cross-domain retrieval, and OOD construction on CodeAlpaca; (b) offline construction cost on LongBench-v2.

lightweight, consistent with the stable LongBench-v2 speedups in Table 4. Beyond drafting, *SENSE* remains efficient during verification through its Soft-gated Evaluation mechanism, improving candidate utilization under a single target-model pass.

5 Conclusion

We present ***SENSE***, a novel retrieval-based speculative decoding framework anchored in semantic alignment to overcome lexical rigidity. To realize this philosophy, *SENSE* leverages Semantic Embedding Navigation (*SEN*) to augment retrieval diversity, and Soft-gated Evaluation (*SE*) to enable tolerance-aware verification while preserving comparable quality under evaluated-setting. Extensive and empirical evaluations across diverse benchmarks and backbones substantiate the effectiveness and robustness of *SENSE*. Furthermore, our analyses suggest two directions for future speculative decoding: semantic-aware verification becomes increasingly important as models make exact-match drafting harder, while retrieval-based drafting remains promising for longer-context inference by avoiding the growing prefix-dependent overhead of draft models.

Limitations

Despite achieving promising results, *SENSE* presents several notable limitations that merit discussion. First, the quality and construction overhead of the datastore constitute critical bottlenecks. The method's effectiveness is fundamentally bounded by the availability of high-quality candidate continuations; constructing comprehensive datastores for new domains incurs non-trivial computational costs that must be amortized across many inference queries. This dependency renders *SENSE* less flexible for rapidly evolving or specialized domains where datastore updates are frequent.

Second, performance varies across model families and scales. Weaker models such as Llama2-7B face calibration challenges, making entropy-guided verification less reliable due to poor probability estimates. Gains also diminish on larger, more capable models, where exact-match constraints are less restrictive under greater semantic diversity. This suggests that *SENSE* is best suited to mid-range models rather than the full spectrum of model sizes.

Third, the entropy-guided verification relies on well-calibrated model confidence scores, which may not hold universally across diverse architectures and training paradigms. The binary gating mechanism (B_{ent} and B_{topk}) requires careful hyperparameter tuning to balance semantic flexibility with output fidelity. Our current evaluation is limited to greedy decoding scenarios; extending *SENSE* to other generation modes would require substantial methodological adaptations. Finally, while our framework decomposes speculative decoding into atomic primitives, the unified testbed itself introduces implementation overhead that may obscure the true efficiency characteristics of individual methods in production settings.

Ethical Considerations

This work focuses on accelerating the inference of existing large language models through retrieval-based speculative decoding, without modifying the models' training data or parameters. Retrieved continuations are used only as speculative proposals and are not incorporated into the target model's conditioning context. Our method inherits the ethical considerations associated with the underlying target models. We do not introduce new risks beyond those already present in standard LLM deployment. All experiments were conducted on publicly available benchmarks using open-source models.

References

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. 2025. Judge Decoding: Faster Speculative Sampling Requires Going Beyond Model Alignment. *arXiv preprint*. ArXiv:2501.19309 [cs].

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks. *arXiv preprint*. ArXiv:2412.15204 [cs].

Sahil Chaudhary. 2026. sahil280114/codealpaca. Original-date: 2023-03-22T11:28:54Z.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint*. ArXiv:2204.02311 [cs].

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 [cs].

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. *arXiv preprint*. ArXiv:2305.14233 [cs].

Maxim Divilkovskiy, Vitaly Malygin, Sergey Zlobin, Stanislav Ilyushin, Sultan Isali, Vasily Kalugin, Nuriza Aitassova, Fei Yi, and Weidi Zeng. 2025. READER: Retrieval-Assisted Drafter for Efficient LLM Inference. *arXiv preprint*. ArXiv:2508.09072 [cs].

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. 2025. Autojudge: Judge decoding without manual annotation. *Preprint*, arXiv:2504.20039.

Milan Gritta, Huiyin Xue, and Gerasimos Lampouras. 2025. DReSD: Dense Retrieval for Speculative Decoding. *arXiv preprint*. ArXiv:2502.15572 [cs].

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang,

Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. 2024. REST: Retrieval-Based Speculative Decoding. *arXiv preprint*. ArXiv:2311.08252 [cs].

Sophia Ho, Jinsol Park, and Patrick Wang. 2024. CREST: Effectively Compacting a Datastore For Retrieval-Based Speculative Decoding. *arXiv preprint*. ArXiv:2408.04678 [cs].

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *Preprint*, arXiv:1904.09751.

Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. 2025. Speculative Decoding and Beyond: An In-Depth Survey of Techniques. *arXiv preprint*. ArXiv:2502.19732 [cs].

Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. 2024. Sam decoding: Speculative decoding via suffix automaton. *Preprint*, arXiv:2411.10666.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv preprint*. ArXiv:1705.03551 [cs].

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. *arXiv preprint*. ArXiv:2211.17192 [cs].

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith C. H. Ngai, and Emad Barsoum. 2025a. Training-Free Loosely Speculative Decoding: Accepting Semantically Correct Drafts Beyond Exact Match. *arXiv preprint*. ArXiv:2511.22972 [cs].

Ying Li, Zhaode Wang, Zhiwen Chen, chengfei lv, and Huan Wang. 2026. ARC-decode: Risk-bounded acceptance for sampling-based speculative decoding.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025b. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. *arXiv preprint*. ArXiv:2401.15077 [cs].

Tianyu Liu, Qitan Lv, Hao Li, Xing Gao, Xiao Sun, and Xiaoyan Sun. 2026. LogitSpec: Accelerating Retrieval-based Speculative Decoding via Next Next Token Speculation. *arXiv preprint*. ArXiv:2507.01449 [cs].

Yu A. Malkov and D. A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv preprint*. ArXiv:1603.09320 [cs].

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949. ACM.

Hyun Ryu and Eric Kim. 2024. Closer Look at Efficient Inference Methods: A Survey of Speculative Decoding. *arXiv preprint*. ArXiv:2411.13157 [cs].

Shwetha Somasundaram, Anirudh Phukan, and Apoorv Saxena. 2024. PLD+: Accelerating LLM inference by leveraging Language Model Artifacts. *arXiv preprint*. ArXiv:2412.01447 [cs].

Lawrence Stewart, Matthew Trager, Sujan Kumar Gonugondla, and Stefano Soatto. 2024. The N-Grammys: Accelerating Autoregressive Inference with Learning-Free Batched Speculation. *arXiv preprint*. ArXiv:2411.03786 [cs].

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint*. ArXiv:2307.09288 [cs].

Jikai Wang, Zhenxu Tian, Juntao Li, Qingrong Xia, Xinyu Duan, Zhefeng Wang, Baoxing Huai, and Min Zhang. 2025. Alignment-Augmented Speculative Decoding with Alignment Sampling and Conditional Verification. *arXiv preprint*. ArXiv:2505.13204 [cs] version: 2.

Zixiao Ken Wang. 2024. Microllama: A 300m-parameter language model trained from scratch. https://github.com/keeeeenw/MicroLlama, https://huggingface.co/keeeeenw/MicroLlama. GitHub and Hugging Face repositories.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with Reference: Lossless Acceleration of Large Language Models. *arXiv preprint*. ArXiv:2304.04487 [cs].

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130B: An Open Bilingual Pre-trained Model. *arXiv preprint*. ArXiv:2210.02414 [cs].

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A survey on efficient inference for large language models. *Preprint*, arXiv:2404.14294.

A Outline

This appendix provides supplementary materials to complement the main text:

- Section B describes the usage of large language models during manuscript preparation.
- Section C presents the detailed algorithmic specifications of the *SENSE* framework.
- Section D provides extended experimental analyses that go beyond the main-text results. It includes an ID/OOD boundary-condition study, a complete verification-strategy comparison, a comprehensive hyperparameter sweep, an accuracy-preservation analysis, an LLM-as-Judge evaluation with per-dimension breakdown, a representative case study illustrating output fidelity under acceleration, a failure-case analysis, detailed datastore resource statistics, and an in-depth *datastore* investigation covering (i) size-scaling behavior and (ii) cross-domain mismatch between task and datastore.
- Section E offers comprehensive experimental details, covering dataset statistics, prompt templates, baseline descriptions, implementation hyperparameters, and hardware configurations.
- Section F outlines promising directions for future research.

B LLM Usage

The large language model (LLM) was utilized as a writing assistant during the preparation of this manuscript. Its application was strictly limited to improving the clarity and grammatical accuracy of the text. Specific uses included rephrasing sentences for better flow and translating initial concepts and drafts from Chinese to English. All core scientific contributions, including the conceptualization of our ***SENSE*** framework, the design of the methodology and experiments, and the analysis and interpretation of the results, are solely the work of the authors. The authors take full responsibility for all claims and the final content of this paper.

C Algorithm Details

This section provides the complete algorithmic specification of the *SENSE* framework. We present four algorithms: the main speculative decoding loop (Algorithm 1), the Semantic Embedding Navigation module (Algorithm 2), the Loose Trie construction (Algorithm 3), and the Soft-gated Evaluation mechanism (Algorithm 4).

C.1 Main Decoding Loop

Algorithm 1 presents the main speculative decoding loop. The framework operates in three phases per iteration: (1) draft generation via semantic retrieval, (2) parallel verification with tree-structured attention, and (3) token acceptance with state updates.

The prefill phase (lines 2–6) initializes the generation by encoding the prompt and populating the dynamic retrieval cache. Each decoding iteration consists of: draft generation (lines 9–10) using Algorithm 2, tree-structured verification (lines 12–15) with a single forward pass over the flattened candidates, and state update (lines 17–24) where accepted tokens are appended to the output sequence.

C.2 Semantic Embedding Navigation

Algorithm 2 describes the draft generation module. It retrieves candidate continuations from both a static corpus and a dynamic in-context cache, then organizes them into a prefix-compressed tree structure for efficient parallel verification.

The module operates in four steps:

Algorithm 1 SENSE Framework

Require: Prompt q, target model $\mathcal{M}$, retriever $\mathcal{R}$, max length T
Ensure: Generated sequence $\mathbf{y}$

1: $\mathbf{y} \leftarrow \textsc{Tokenize}(q)$
2: $h, p, \mathcal{K} \leftarrow \mathcal{M}(\mathbf{y})$
3: $\mathcal{R}.\textsc{Update}(h, \mathbf{y})$
4: **while** $|\mathbf{y}| < T$ **and** $\mathbf{y}_{-1} \neq \texttt{EOS}$ **do**
5: $\quad \tilde{x} \leftarrow \arg\max p$
6: $\quad D, \mathcal{T}_{\text{flat}}, A_{\text{tree}}, \Gamma \leftarrow \text{SEN}(h, \tilde{x}, \mathcal{R})$
7: $\quad A_{\text{full}} \leftarrow \textsc{TreeMask}(A_{\text{tree}}, |\mathbf{y}|, |\mathcal{K}|)$
8: $\quad H_{\text{flat}}, P_{\text{flat}} \leftarrow \mathcal{M}.\textsc{Forward}(\mathcal{T}_{\text{flat}}, A_{\text{full}}, \mathcal{K})$
9: $\quad H^{n,0}_{\text{align}}, P^{n,0}_{\text{align}} \leftarrow h, p \quad \forall n$
10: $\quad H^{n,j}_{\text{align}}, P^{n,j}_{\text{align}} \leftarrow \Gamma(H_{\text{flat}})^{n,j}, \Gamma(P_{\text{flat}})^{n,j} \quad \forall n,\ j \geq 1$
11: $\quad n^*, L^* \leftarrow \text{SE}(D, P_{\text{align}})$
12: $\quad$ **if** $L^* = 0$ **then**
13: $\quad\quad \mathbf{a} \leftarrow [\tilde{x}]$
14: $\quad\quad \mathbf{y} \leftarrow \mathbf{y} \| \mathbf{a}$
15: $\quad\quad h, p, \mathcal{K} \leftarrow \mathcal{M}.\textsc{Step}(\tilde{x}, \mathcal{K})$
16: $\quad\quad \mathcal{R}.\textsc{Update}(h, \mathbf{a})$
17: $\quad$ **else**
18: $\quad\quad \mathbf{a} \leftarrow D_{n^*, 1:L^*}$
19: $\quad\quad \mathbf{y} \leftarrow \mathbf{y} \| \mathbf{a}$
20: $\quad\quad h \leftarrow H^{n^*, L^*}_{\text{align}}$
21: $\quad\quad p \leftarrow P^{n^*, L^*}_{\text{align}}$
22: $\quad\quad \mathcal{K} \leftarrow \textsc{UpdateKV}(\mathcal{K}, n^*, L^*, \Gamma)$
23: $\quad\quad \mathcal{R}.\textsc{Update}(H^{n^*, 1:L^*}_{\text{align}}, \mathbf{a})$
24: $\quad$ **end if**
25: **end while**
26: **Return** $\mathbf{y}$

Query Construction (line 2). The hidden state h is projected to a lower-dimensional space via PCA and L2-normalized to form the retrieval query $k \in \mathbb{R}^v$ where $v \ll d$.

Hybrid Retrieval (lines 4–7). Candidates are retrieved from two sources: a static pre-built corpus $\mathcal{C}_s$ via FAISS approximate nearest neighbor search, and a dynamic cache $\mathcal{C}_d$ that stores recent generation context. Dynamic results receive a score boost δ to prioritize in-context continuations.

Composite Scoring (lines 9–14). Each candidate is scored by a weighted combination of lexical matching (whether its first token equals the anchor $\tilde{x}$) and semantic similarity. The constraint $\alpha \gg \beta$ ensures candidates with matching first tokens are strongly preferred. Lines 13–14 guarantee at least two candidates start with the anchor token.

Algorithm 2 Semantic Embedding Navigation (SEN)

Require: Hidden state $h \in \mathbb{R}^d$, anchor token $\tilde{x}$, retriever $\mathcal{R}$
Require: Hyperparameters: N, M, N', α, β where $\alpha \gg \beta$
Ensure: Candidates D, flattened tree $\mathcal{T}$, mask $\mathbf{M}$, mapping Γ

1:
2: $k \leftarrow \text{L2}\textsc{Norm}(\text{PCA}(h))$
3:
4: $D_s, \mathbf{s}_s \leftarrow \text{FAISS}(k, \mathcal{C}_s, N')$
5: $D_d, \mathbf{s}_d \leftarrow \textsc{DynSearch}(h, \mathcal{C}_d, \tilde{x})$
6: $D_{\text{all}} \leftarrow D_s \cup D_d$
7: $\mathbf{s} \leftarrow [\mathbf{s}_s; \mathbf{s}_d + \delta]$
8:
9: **for** $i = 1$ **to** $|D_{\text{all}}|$ **do**
10: $\quad S_i \leftarrow \alpha \cdot \mathbb{I}(D_{\text{all}}[i, 0] = \tilde{x}) + \beta \cdot \mathbf{s}_i$
11: **end for**
12: $D \leftarrow \textsc{TopN}(D_{\text{all}}, S)$
13: $D[0, 0] \leftarrow \tilde{x}$
14: $D[-1, 0] \leftarrow \tilde{x}$
15:
16: $D_{\text{sorted}} \leftarrow \textsc{LexSort}(D)$
17: $\mathcal{T}, \mathbf{M}, \Gamma \leftarrow \textsc{LooseTrie}(D_{\text{sorted}})$
18:
19: **Return** $D, \mathcal{T}, \mathbf{M}, \Gamma$

Trie Construction (lines 16–17). Selected candidates are lexicographically sorted and passed to Algorithm 3 for prefix compression.

C.3 Loose Trie Construction

Algorithm 3 constructs a prefix-compressed tree from sorted candidates. Unlike a canonical trie that requires $O(NM^2)$ pairwise comparisons, our approach exploits lexicographic ordering to achieve $O(NM)$ complexity through adjacent-only comparisons.

The algorithm proceeds as follows:

Prefix Detection (lines 2–5). For each position, we check whether the current candidate matches its predecessor (line 3). The cumulative product along dimension 1 (line 4) propagates this match status: a position is marked as shared prefix only if all preceding positions also match. The negation yields the unique token mask.

Algorithm 3 Loose Trie Construction

Require: Sorted candidates $D \in \mathbb{Z}^{N \times M}$
Ensure: Flattened tree $\mathcal{T}$, attention mask $\mathbf{M}$, mapping Γ

1:
2: $\texttt{match} \leftarrow \textsc{Zeros}(N, M)$
3: $\texttt{match}[1:] \leftarrow (D[1:] = D[:-1])$
4: $\texttt{is_prefix} \leftarrow \textsc{CumProd}(\texttt{match}, \text{dim} = 1)$
5: $\texttt{unique} \leftarrow \neg\, \texttt{is_prefix}$
6:
7: $\mathcal{T} \leftarrow D[\texttt{unique}]$
8: $L \leftarrow |\mathcal{T}|$
9:
10: $\texttt{idx} \leftarrow \textsc{CumSum}(\texttt{unique}.\textsc{Flat}()) - 1$
11: $\Gamma \leftarrow \textsc{Reshape}(\texttt{idx}, (N, M))$
12: $\texttt{tmp} \leftarrow \textsc{Where}(\texttt{unique}, \Gamma, -1)$
13: $\Gamma \leftarrow \textsc{CumMax}(\texttt{tmp}, \text{dim} = 0)$
14:
15: $\mathbf{M} \leftarrow \textsc{Zeros}(L, L)$
16: $\mathbf{I}, \mathbf{J} \leftarrow \textsc{TrilIndices}(M, M)$
17: **for** $i = 1$ **to** N **do**
18: $\quad \mathbf{M}[\Gamma[i, \mathbf{I}], \Gamma[i, \mathbf{J}]] \leftarrow 1$
19: **end for**
20:
21: **Return** $\mathcal{T}, \mathbf{M}, \Gamma$

Algorithm 4 Soft-gated Evaluation (SE)

Require: Aligned candidate tokens $X \in \mathbb{Z}^{N \times M}$
Require: Aligned target distributions $P^{n,j}_{\text{align}}$ for $n = 1, \ldots, N$ and $j = 0, \ldots, M-1$
Require: Hyperparameters: entropy threshold γ, top-k, window size W, mismatch threshold ρ
Ensure: Selected draft n^*, accepted prefix length L^*

1: $Y_{n,j} \leftarrow \arg\max_{v \in \mathcal{V}} P^{n,j}_{\text{align}}(v)$
2: $B^{n,j}_{\text{em}} \leftarrow \mathbb{I}(X_{n,j} = Y_{n,j})$
3: $E_{n,j} \leftarrow -\sum_{v \in \mathcal{V}} P^{n,j}_{\text{align}}(v) \log P^{n,j}_{\text{align}}(v)$
4: $B^{n,j}_{\text{ent}} \leftarrow \mathbb{I}(E_{n,j} > \gamma)$
5: $\mathcal{Y}^{n,j}_{\text{topk}} \leftarrow \textsc{TopK}_k(P^{n,j}_{\text{align}})$
6: $B^{n,j}_{\text{topk}} \leftarrow \mathbb{I}(X_{n,j} \in \mathcal{Y}^{n,j}_{\text{topk}})$
7: $\bar{B}_{\text{em}} \leftarrow \mathbf{1} - B_{\text{em}}$
8: $\bar{B}_{\text{pad}} \leftarrow \textsc{Pad}(\bar{B}_{\text{em}}, (0, W), 1)$
9: $R \leftarrow \textsc{Conv1D}(\bar{B}_{\text{pad}}, \mathbf{1}_W)[:, 0:M]$
10: $B_{\text{con}} \leftarrow \mathbb{I}(R \leq \rho)$
11: $G \leftarrow B_{\text{em}} \vee \big(B_{\text{ent}} \wedge (B_{\text{topk}} \vee B_{\text{con}})\big)$
12: $G_{\text{seq}} \leftarrow \textsc{CumProd}(G, \text{dim} = 1)$
13: $L_n \leftarrow \sum_{j=0}^{M-1} G_{\text{seq}}[n, j]$
14: $n^* \leftarrow \arg\max_{n \in \{1, \ldots, N\}} L_n$
15: $L^* \leftarrow L_{n^*}$
16: **Return** n^*, L^*

Tree Extraction (lines 7–8). Tokens marked as unique are extracted to form the flattened tree $\mathcal{T}$, eliminating redundant prefix tokens.

Index Mapping (lines 10–13). The gather-scatter mapping Γ records which tree position each candidate token maps to. For shared prefixes, indices are propagated from predecessor candidates via cumulative maximum.

Attention Mask (lines 15–19). The tree attention mask $\mathbf{M}$ encodes causal visibility: within each candidate path, earlier positions can be attended to by later positions, following standard causal attention semantics.

C.4 Soft-gated Evaluation

Algorithm 4 performs verification through a cascade of four binary masks. It operates only on the M candidate positions; the lookahead output at position M is reserved for the decoding-state update in Algorithm 1. This design allows controlled relaxed acceptance of target-supported tokens at uncertain positions. The four masks capture different acceptance criteria:

Exact Match B_{em} (line 4). The baseline criterion: accept if the draft token exactly matches the greedy prediction.

Entropy Gate B_{ent} (lines 6–8). Identifies high-uncertainty positions where the model's probability mass is spread across multiple tokens. Entropy above threshold γ identifies positions where the model assigns probability mass to multiple continuations.

Top-k Relaxation B_{topk} (lines 10–11). For uncertain positions, accept tokens within the top-k predictions. This constrains relaxed acceptance to alternative tokens with high target-model support.

Convolutional Lookahead B_{con} (lines 13–15). A robustness check examining whether mismatches are isolated or clustered. The 1D convolution counts mismatches in a sliding window of size W. Isolated mismatches (zero future mismatches) are more likely benign.

Final Decision (line 17). A token is accepted if it matches exactly, *or* if the position has high uncertainty *and* the token satisfies either the top-k or

lookahead criterion. The cumulative product (line 19) ensures only contiguous prefixes are accepted.

C.5 Complexity Analysis

Let N denote the number of draft candidates, M the maximum draft length, and $|\mathcal{C}_s|$ the static corpus size.

- **Retrieval:** $O(\log |\mathcal{C}_s|)$ for FAISS search, $O(|\mathcal{C}_d|)$ for dynamic cache lookup.
- **Trie Construction:** $O(NM)$ via vectorized adjacent comparisons.
- **Verification:** $O(NM)$ through fully parallelized mask operations.
- **Model Forward:** Single pass over $L \leq NM$ flattened tree tokens.

The dominant cost per iteration is the target model forward pass, whose complexity is reduced from $O(NM)$ separate calls to a single batched call through tree-structured attention.

C.6 Tree Attention Mask for Static KV Cache

Given the draft attention mask $\mathbf{M} \in \{0,1\}^{L \times L}$, current sequence length m, and maximum cache length M_{cache}, we construct:

$$\mathbf{M}_{\text{full}}[i,j] = \begin{cases} 0 & \text{if } j < m, \\ 0 & \text{if } \substack{m \leq j < m+L \\ \text{and } \mathbf{M}[i,j-m]=1}, \\ -\infty & \text{otherwise.} \end{cases} \quad (14)$$

D Further Analysis

D.1 Analysis on ID/OOD Boundary Conditions

The inversion observed in Llama2-7B (where OOD outperforms ID) highlights a critical boundary condition driven by model confidence.

- Low-Entropy Regime (Strong Models): Models like Qwen3 are highly confident (low entropy). Any deviation in the OOD datastore is treated as a distribution shift and rejected. Thus, ID (distributional alignment) is essential.
- High-Entropy Regime (Weak Models): Models like Llama2-7B exhibit high uncertainty and generation noise. In this regime, ID datastores merely mimic the model's own hallucinations. OOD datastores, derived from ground truth, provide better semantic guidance. SENSE's entropy-gating mechanism is uniquely designed to bypass exact matching in these high-uncertainty regions, allowing the model to accept these "corrective" OOD drafts.

D.2 Verification Strategy Comparison

To provide a comprehensive view of verification strategy performance, Table 7 presents the complete comparison across all model-dataset combinations. We fix the SEN retrieval module and compare different verification methods: ARC, FLY, and our Soft-gated Evaluation (SE). The table reports both speedup ratios and mean accepted tokens (τ) for each configuration.

The superiority of SE stems from its synergistic masking mechanism $G = B_{em} \vee (B_{ent} \wedge (B_{topk} \vee B_{con}))$.

Unlike ARC, which rejects candidates solely based on a rigid risk bound, or FLY, which relies on a strictly linear lookahead, SE introduces a *dual-pathway rescue mechanism*:

1. **Distributional Rescue (B_{topk}):** For high-entropy positions where the model is uncertain (e.g., synonyms like "glad" vs. "happy"), SE accepts the token if it resides within the top-k probability mass. This captures semantic equivalence that exact-match or strict risk bounds often miss.
2. **Structural Rescue (B_{con}):** For positions with isolated lexical mismatches (e.g., singular/plural variations), the convolutional mask (B_{con}) aggregates local error density. If the mismatch is an isolated event within a correct neighborhood, the token is retained.

This combination allows SE to salvage valid drafts that baselines would erroneously prune, directly translating to higher acceptance rates and speedups.

The Calibration Boundary Condition (Llama2-7B).

As noted in Section D.2, baselines exhibit a marginal lead on Llama2-7B (e.g., 6.00× vs. 5.85×). While Section D.1 attributes the success of OOD datastores to the *generation quality* of weak models, the verification performance is governed by *model calibration*.

Entropy-guided verification operates on the assumption that *predicted uncertainty reflects se-*

mantic flexibility. However, smaller, older models like Llama2-7B often suffer from poor calibration—they may exhibit "confident errors" (low entropy on wrong tokens) or "diffuse correctness" (high entropy on the correct token but poor top-k clustering).

In such regimes, the gating signal B_{ent} becomes noisy, occasionally triggering relaxed verification when it should not, or failing to trigger it when appropriate. In contrast, modern architectures (Qwen series) demonstrate superior calibration, making entropy a reliable proxy for semantic verification and enabling SE to reach its full potential.

D.3 Comprehensive Hyperparameter Study

To provide a thorough understanding of SENSE's sensitivity to hyperparameter choices, we conduct an exhaustive grid search over the three key parameters in the Soft-gated Evaluation module: entropy threshold (θ_e), Top-k value, and convolution window size (w). All experiments are conducted on Qwen2.5-7B using the GSM8K benchmark with the ID datastore.

Parameter Ranges. We evaluate:

- Entropy threshold $\theta_e \in \{0.01, 0.05, 0.10, 0.30\}$
- Top-$k \in \{2, 3, 4, 10\}$
- Window size $w \in \{5, 6, 7, 8, 9\}$

This yields $4 \times 4 \times 5 = 80$ configurations. Complete results are presented in Table 8.

Key Observations. (1) **Entropy threshold** acts as the primary gating mechanism: higher values ($\theta_e = 0.30$) restrict relaxed verification to only the most uncertain positions, yielding conservative but stable performance; lower values ($\theta_e = 0.01$) enable more aggressive acceptance, improving speedup but requiring careful tuning of downstream criteria. (2) **Top-k** controls the distributional tolerance: smaller k values enforce stricter semantic alignment, while larger values ($k = 10$) risk accepting low-probability tokens that may degrade accuracy. (3) **Window size** affects robustness to isolated mismatches: larger windows provide more context for the convolutional lookahead but may over-smooth the acceptance signal.

D.4 Accuracy Preservation Analysis

To validate that SENSE preserves generation quality while achieving speedup, we conduct a comprehensive accuracy ablation study across multiple model-dataset configurations. Table 9 reports the task-specific metrics for vanilla (unaccelerated) decoding versus SENSE with both OOD and ID datastores. For GSM8K and TriviaQA, we report exact-match accuracy (ACC); for CodeAlpaca and UltraChat, we report ROUGE-L scores to assess generation quality.

Key Observations. The results demonstrate that SENSE largely preserves the generation quality of the target model across diverse tasks.

1. **Mathematical Reasoning (GSM8K):** Accuracy degradation is minimal, with SENSE configurations achieving 0.89–0.93 compared to vanilla's 0.92–0.95. The mean absolute accuracy drop is only 2.3%, indicating that the soft-gated verification mechanism effectively maintains reasoning integrity.

2. **Code Generation (CodeAlpaca):** SENSE-OOD occasionally outperforms vanilla (e.g., 0.443 vs. 0.427 on Qwen3-8B), suggesting that high-quality retrieved drafts can provide beneficial semantic guidance. However, Qwen3-14B shows degraded ROUGE-L scores for both SENSE variants, warranting further investigation.

3. **Dialogue (UltraChat):** ROUGE-L scores remain stable across most configurations, with variations typically within 0.02 of vanilla baselines.

4. **Question Answering (TriviaQA):** Performance remains robust under the SENSE method as measured by the ACC metric.

Overall, these results validate that SENSE achieves substantial speedup (2–3$\times$ as shown in Table 1) while maintaining task performance within acceptable margins for most configurations. The observed trade-off aligns with the hyperparameter sensitivity analysis in Table 8, where our default configuration (θ_e=0.05, k=3) balances acceleration with quality preservation.

D.5 Tree-Structured Parallelism Analysis.

As illustrated in Figure 6, the Tree-Structured Parallelism in Section 3.2 maintains robust storage efficiency, achieving an average compression ratio of **73.4**%. Notably, compression efficiency exhibits an inverse correlation with model scale. This trend stems from the richer lexical diversity of larger models, where enhanced generative capabilities

Table 7: Complete performance comparison of verification strategies. We fix the SEN retrieval module and compare different verification methods: ARC, FLY, and our Soft-gated Evaluation (SE). **Bold** indicates the best performance in each row.

Model	Method	GSM8K		CodeAlpaca		UltraChat		TriviaQA		Mean	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Qwen2.5-7B	ARC	2.51	3.09	2.87	3.71	2.14	2.71	1.97	2.42	2.37	2.98
	FLY	2.43	3.02	2.83	3.71	2.02	2.57	2.32	2.89	2.40	3.05
	Ours	**2.94**	**3.44**	**3.30**	**3.95**	**2.59**	**3.04**	**2.72**	**3.63**	**2.89**	**3.52**
Qwen2.5-14B	ARC	2.66	3.26	**2.49**	3.12	1.89	1.33	1.85	2.28	2.22	2.50
	FLY	2.51	3.10	**2.49**	**3.14**	1.77	2.20	1.87	2.32	2.16	2.69
	Ours	**3.00**	**3.48**	2.48	2.90	**2.40**	**2.98**	**2.47**	**2.56**	**2.59**	**2.96**
Llama2-7B	ARC	6.73	7.00	6.34	5.17	**8.52**	**9.08**	2.41	2.20	6.00	5.86
	FLY	**7.40**	**7.82**	**6.92**	**5.73**	6.78	7.09	2.88	2.62	**6.00**	5.82
	Ours	5.45	5.80	6.86	5.63	3.28	3.44	**7.46**	**7.78**	5.76	**5.85**
Llama2-13B	ARC	1.14	2.29	2.60	4.79	1.22	2.31	4.89	8.94	2.46	4.58
	FLY	1.10	2.26	3.36	6.19	1.49	**2.83**	4.81	8.75	2.69	5.01
	Ours	**1.24**	**2.44**	**3.72**	**6.83**	**1.45**	2.73	**5.36**	**9.75**	**2.94**	**5.44**

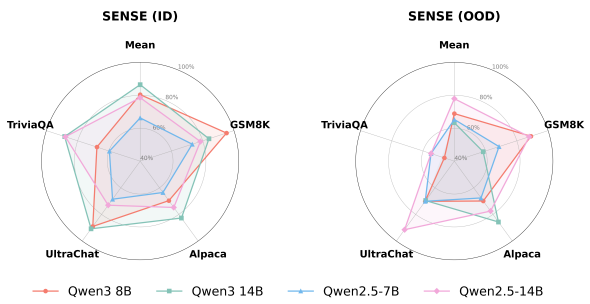

Figure 6: Compression efficiency analysis of the Tree-Structured Parallelism across diverse model scales and benchmarks.

naturally diminish the local token redundancy required for effective compaction.

D.6 LLM-as-Judge Evaluation Details

This section provides the full protocol and per-dimension breakdown for the LLM-as-Judge evaluation. The goal of this evaluation is to complement the task-metric preservation analysis (Section D.4) with a *qualitative* assessment, directly addressing the concern that relaxed verification may silently trade quality for speed.

Protocol. We adopt Claude Opus 4.6 as an independent judge model. For each prompt, the judge is shown the original input together with an anonymized model output (method identity blinded and order randomized) and is asked to score the response along three orthogonal dimensions on a 1–5 Likert scale:

- **Correctness**—factual accuracy and logical validity of the answer;
- **Completeness**—coverage of the task requirements and instruction following;
- **Quality**—writing fluency, coherence, and reasoning clarity.

Evaluation is performed on four representative tasks—GSM8K (math reasoning), CodeAlpaca (code generation), UltraChat (open-ended dialogue), and TriviaQA (knowledge QA)—under the *same* fixed-*SEN* retrieval setting as Figure 3 (Qwen2.5-7B as the target model). This setting isolates the effect of the verifier: the only variable across SEN + ARC, SEN + FLY, and *SENSE* (SEN + SE) is the acceptance rule.

Per-dimension Results. Table 10 reports the mean and standard deviation of each dimension. The three verifiers track each other closely: overall scores are ARC 3.44, FLY 3.34, and *SENSE* 3.39, with per-dimension gaps (Correctness 3.79–3.90, Completeness 3.36–3.43, Quality 2.88–2.98) all below 0.12 and well within one standard deviation. *SENSE* is on par with ARC and above FLY on average, confirming that soft-gated relaxation is quality-neutral relative to both baselines. Combined with the 20–22% speedup advantage in Figure 3, *SE* therefore dominates on the quality–speed Pareto frontier.

Judge Prompt Template. The judge is invoked once per (prompt, response) pair with the following template. The evaluator is instructed to rate the response on three dimensions using integers from 1 (very poor) to 5 (excellent): *Correctness* (factual / logical accuracy of the answer), *Completeness* (coverage of what the prompt requests), and *Quality* (fluency, coherence, and clarity of reasoning). The judge returns a JSON object of the form `{"correctness": int,`

Table 8: Comprehensive hyperparameter study on Qwen2.5-7B (GSM8K, ID datastore). We report Accuracy (%), Speedup ratio, and mean accepted tokens (τ) across 80 configurations of entropy threshold θ_e, Top-k, and window size w. Highlighted cells indicate our default configuration (θ_e=0.05, k=3, w=6).

		$k = 2$			$k = 3$			$k = 4$			$k = 10$		
θ_e	w	Acc	SpeedUp	τ	Acc	SpeedUp	τ	Acc	SpeedUp	τ	Acc	SpeedUp	τ
0.01	5	0.89	3.28	3.93	0.76	5.02	5.70	0.67	5.95	6.50	0.15	8.12	8.28
	6	0.90	3.25	3.85	0.82	4.30	4.99	0.77	5.59	6.14	0.12	8.33	8.46
	7	0.90	3.07	3.61	0.89	3.89	4.53	0.73	5.09	5.64	0.16	8.16	8.35
	8	0.92	3.00	3.55	0.88	3.93	4.57	0.71	5.23	5.79	0.18	8.12	8.31
	9	0.93	2.95	3.51	0.88	3.96	4.59	0.72	5.60	6.15	0.18	8.10	8.27
0.05	5	0.99	2.74	3.36	0.96	2.98	3.59	0.87	3.19	3.85	0.70	5.81	6.39
	6	0.97	2.70	3.29	0.98	2.82	3.44	0.88	3.42	4.10	0.73	5.62	6.19
	7	0.97	2.67	3.27	0.97	2.79	3.40	0.89	3.14	3.77	0.74	5.28	5.91
	8	0.97	2.67	3.25	0.97	2.79	3.39	0.89	3.16	3.75	0.76	5.28	5.93
	9	0.97	2.65	3.23	0.98	2.75	3.37	0.90	3.17	3.75	0.78	5.09	5.73
0.10	5	0.95	2.37	2.95	0.95	2.41	2.98	0.95	2.42	3.00	0.90	2.48	3.06
	6	0.96	2.36	2.93	0.95	2.39	2.97	0.96	2.43	3.00	0.91	2.46	3.06
	7	0.96	2.35	2.92	0.95	2.40	2.96	0.97	2.41	2.98	0.91	2.47	3.06
	8	0.96	2.36	2.92	0.95	2.40	2.96	0.98	2.40	2.97	0.91	2.47	3.05
	9	0.96	2.35	2.91	0.95	2.38	2.94	0.98	2.39	2.97	0.91	2.46	3.05
0.30	5	0.93	2.24	2.78	0.93	2.25	2.78	0.93	2.24	2.78	0.93	2.24	2.78
	6	0.93	2.25	2.78	0.93	2.25	2.78	0.93	2.25	2.78	0.93	2.24	2.78
	7	0.93	2.24	2.78	0.93	2.23	2.78	0.93	2.26	2.78	0.93	2.26	2.78
	8	0.93	2.24	2.78	0.93	2.26	2.78	0.93	2.24	2.78	0.93	2.25	2.78
	9	0.93	2.25	2.78	0.93	2.26	2.78	0.93	2.25	2.78	0.93	2.24	2.78

Table 9: Accuracy preservation analysis across models and datasets. We compare vanilla decoding (baseline) against SENSE-OOD and SENSE-ID configurations. GSM8K and TriviaQA report accuracy (ACC); CodeAlpaca and UltraChat report ROUGE-L scores. **Bold** indicates highest value per model-dataset pair.

Model	Method	GSM8K	CodeAlpaca	UltraChat	TriviaQA
		ACC	ROUGE-L	ROUGE-L	ACC
Qwen3-8B	Vanilla	**0.93**	0.427	**0.206**	**0.45**
	SENSE-OOD	0.91	**0.443**	0.189	0.43
	SENSE-ID	0.90	0.435	0.188	0.42
Qwen3-14B	Vanilla	0.92	0.294	0.204	**0.56**
	SENSE-OOD	0.92	0.288	**0.205**	**0.56**
	SENSE-ID	**0.93**	**0.303**	0.196	**0.56**
Qwen2.5-7B	Vanilla	**0.92**	0.402	**0.221**	**0.48**
	SENSE-OOD	**0.92**	0.395	0.200	0.47
	SENSE-ID	0.90	**0.406**	0.213	0.44
Qwen2.5-14B	Vanilla	**0.95**	0.416	**0.227**	0.66
	SENSE-OOD	**0.95**	**0.425**	**0.227**	0.66
	SENSE-ID	0.89	**0.425**	0.207	**0.68**

Table 10: Full LLM-as-Judge breakdown on Qwen2.5-7B under a fixed *SEN* retriever. Scores are mean ± std on a 1–5 scale; *Avg* aggregates the three dimensions per row.

Task	Method	Correctness	Completeness	Quality	Avg
CodeAlpaca	SEN + ARC	4.33±1.07	4.12±1.21	3.97±1.14	4.14±1.08
	SEN + FLY	4.20±1.18	4.04±1.29	3.87±1.25	4.04±1.17
	SENSE	4.27±1.07	4.11±1.22	3.92±1.15	4.10±1.06
GSM8K	SEN + ARC	4.68±1.09	2.51±0.62	1.66±0.47	2.95±0.63
	SEN + FLY	4.72±1.02	2.55±0.62	1.68±0.47	2.98±0.61
	SENSE	4.72±1.02	2.55±0.61	1.66±0.47	2.98±0.60
TriviaQA	SEN + ARC	2.81±1.87	3.02±1.36	2.41±1.09	2.75±1.38
	SEN + FLY	2.62±1.89	2.94±1.33	2.30±1.11	2.62±1.39
	SENSE	2.77±1.98	3.02±1.78	2.62±1.64	2.80±1.77
UltraChat	SEN + ARC	3.79±0.99	4.08±0.98	3.87±0.84	3.91±0.86
	SEN + FLY	3.60±1.06	3.90±1.02	3.66±0.97	3.72±0.92
	SENSE	3.58±1.08	3.88±1.09	3.56±1.09	3.67±1.00
Overall	SEN + ARC	3.90	3.43	2.98	**3.44**
	SEN + FLY	3.79	3.36	2.88	**3.34**
	SENSE	3.84	3.39	2.94	**3.39**

`"completeness": int, "quality": int, "rationale": str}`, followed by the user prompt and the model response as separate fields. Scores are averaged per (task, method) cell. Method identity is blinded and the presentation order of SEN + ARC, SEN + FLY, and *SENSE* is randomized across items to avoid positional bias.

D.7 Case Study

Figures 7a and 7b present a representative case from the UltraChat dataset using Qwen 2.5 7B, comparing SENSE against the Vanilla baseline. The input prompt requests a comprehensive blog post about the top 10 most eco-friendly cities and their renewable energy initiatives, requiring the model to generate a long-form response exceeding 1,000 words.

As shown in the figures, the outputs generated by SENSE and Vanilla are virtually identical in content, structure, and quality. Both responses cover the same cities (Copenhagen, Freiburg, etc.), present consistent factual information (e.g., carbon

emissions reduction statistics, public transportation coverage), and maintain the same organizational flow throughout the article. This demonstrates that SENSE preserves the generation quality of the target model without introducing any degradation or deviation.

More importantly, SENSE achieves a significant speedup of **2.78**× compared to Vanilla decoding. Specifically, SENSE completes the generation in 59.57 seconds with a throughput of 34.67 tokens per second (TPS), while Vanilla requires 164.92 seconds at only 12.47 TPS—both producing approximately 2,056 output tokens. This case exemplifies the core advantage of our approach: substantial acceleration in inference speed while maintaining output fidelity.

D.8 Failure Case Analysis

Figures 8a and 8b present a failure case from the TriviaQA dataset using Qwen 2.5 7B, comparing *SENSE* against the Vanilla baseline. The question asks: *"Who had a 70s No.1 hit with Kiss You All Over?"* The ground truth answer is **Exile**, an American band whose song *Kiss You All Over* reached No.1 on the Billboard Hot 100 in 1978.

Notably, neither *SENSE* nor Vanilla produces the correct answer. *SENSE* incorrectly attributes the song to *Wham!* (claiming a 1981 release), while Vanilla misidentifies the artist as *Brotherhood of Man*. Both responses exhibit confident yet factually incorrect reasoning chains, a hallmark of *hallucination* in large language models.

We emphasize that this failure case does not reflect a fundamental limitation of our proposed method. Rather, it reveals an inherent constraint of the underlying language model itself. Several observations support this conclusion:

(1) Consistent failure across methods. Both *SENSE* and Vanilla fail on this question, indicating that the error originates from the base model's knowledge gaps rather than from our framework.

(2) Knowledge boundary of the base model. The question requires domain-specific knowledge about 1970s pop music chart history, which may be underrepresented in the model's training corpus. Small models like Qwen 2.5 7B are particularly susceptible to such knowledge gaps, especially for niche trivia spanning specific decades and regions.

(3) Confident hallucination pattern. Both outputs demonstrate a common failure mode where the model generates plausible-sounding but incorrect information with high confidence. The detailed step-by-step reasoning (e.g., "reached No.1 on the UK Singles Chart") creates an illusion of reliability while being factually wrong.

Potential solutions. These knowledge-intensive failures can be effectively mitigated through complementary techniques such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) or real-time web search integration, which provide external knowledge grounding. Such approaches are orthogonal to *SENSE* and can be seamlessly combined with our method to enhance both efficiency and factual accuracy.

Table 11: Overall datastore resource statistics across all 24 configurations (6 models × 4 test sets). All memory values are in megabytes (MB).

Metric	Mean	Min	Max	Std
Total RAM Memory	1770.06	105.63	3690.47	1221.90
Index Size on Disk	459.73	0.00	1087.39	399.87
Metadata Memory	1715.97	73.55	3610.37	1214.92
Cache Storage	54.03	28.02	80.03	22.10
Peak RSS Memory	4577.93	2131.49	7412.49	1630.39
RSS Delta Mean	7.25	1.61	24.15	6.77
Acceptance Rate	0.07	0.02	0.35	0.07

Table 12: Total RAM memory usage (MB) by model and dataset. **Bold** indicates lowest (best) value in each column.

Model	GSM8K	CodeAlpaca	TriviaQA	UltraChat	Mean
Llama2-7B	**105.63**	807.74	1796.68	2647.28	1339.33
Llama2-13B	320.54	**749.01**	**593.77**	**778.22**	**610.38**
Qwen2.5-7B	268.19	2839.18	1436.67	3277.00	1955.26
Qwen2.5-14B	325.25	3690.47	3114.02	3048.90	2544.66
Qwen3-8B	395.49	2575.56	3308.63	2584.98	2216.16
Qwen3-14B	406.60	1945.01	2890.36	2576.31	1954.57

D.9 Datastore Resources Statics

This section provides detailed resource consumption statistics for the *SENSE* framework across all evaluated model-dataset configurations. We report memory usage metrics collected from 24 experimental runs (6 models × 4 test sets). The statistics encompass datastore memory footprint, FAISS index storage, and runtime memory consumption during inference.

Table 11 summarizes the overall resource statistics across all configurations. Table 12 presents the total RAM memory usage broken down by model and dataset. Table 13 provides detailed RSS (Resident Set Size) metrics for each model, capturing runtime memory behavior during speculative decoding.

Input Prompts
<|im_start|>system\n You are a helpful assistant. <|im_end|> \n<|im_start|>user\nWrite a comprehensive blog post of at least 1000 words about the top 10 most eco-friendly cities in the world and their renewable energy initiatives. Use a formal and informative tone, and include statistics, case studies, and expert opinions to support your claims. Make sure to cover various aspects of sustainability, such as public transportation, waste management, green buildings, and carbon emissions reduction. Additionally, provide specific examples of innovative renewable energy projects implemented in each city and assess their impact on the environment and the local community. Provide credible sources and links to relevant articles, reports, and websites to enhance the article's credibility and appeal to the readers. <|im_end|>\n<|im_start|>assistant\n

(a) Case Study Prompts

SENSE	Vanilla
### Top 10 Most Eco-Friendly Cities in the World: Renewable Energy Initiatives and Sustainability Practices\n\nIn an era where environmental sustainability is paramount, cities around the globe are increasingly adopting innovative strategies to reduce their carbon footprints and promote green living. This blog post explores the top 10 most eco-friendly cities in the world, highlighting their renewable energy initiatives and other sustainable practices. From cutting-edge public transportation systems to advanced waste management solutions, these cities are leading the way in creating a more sustainable future.\n\n	### Top 10 Most Eco-Friendly Cities in the World: Renewable Energy Initiatives and Sustainability Practices\n\nIn an era where environmental sustainability is paramount, cities around the globe are increasingly adopting innovative strategies to reduce their carbon footprints and promote green living. This blog post explores the top 10 most eco-friendly cities in the world, highlighting their renewable energy initiatives and other sustainable practices. From cutting-edge public transportation systems to advanced waste management solutions, these cities are leading the way in creating a more sustainable future.\n\n
#### 1. Copenhagen, Denmark\n\nCopenhagen is often cited as one of the world's most sustainable cities, with a strong focus on renewable energy and green transportation. The city aims to be carbon neutral by 2025 and has made significant strides in this direction. According to the Global Green Economy Index, Copenhagen ranks first in the world for green economy performance.	#### 1. Copenhagen, Denmark\n\nCopenhagen is often cited as one of the world's most sustainable cities, with a strong focus on renewable energy and green transportation. The city aims to be carbon neutral by 2025 and has made significant strides in this direction. According to the Global Green Economy Index, Copenhagen ranks first in the world for green economy performance.
.....	.....
#### 3. Freiburg, Germany\n\nFreiburg is a pioneer in sustainable urban planning and has been at the forefront of renewable energy initiatives for decades. The city's commitment to sustainability is evident in its green buildings, public transportation, and waste management systems.\n\n**Renewable Energy Initiatives:**\n- **Solar Power:** Freiburg has one of the highest percentages of solar panels per capita in the world.\n- **Wind Power:** The city has several wind farms and is expanding its renewable energy capacity.\n\n**Impact:**\n- **Carbon Emissions:** Freiburg has reduced its carbon emissions by 25% since 1990.\n- **Public Transportation:** The city has an extensive network of buses and trams, with over 70% of residents using public transportation.\n\n	#### 3. Freiburg, Germany\n\nFreiburg is a pioneer in sustainable urban planning and has been at the forefront of renewable energy initiatives for decades. The city's commitment to sustainability is evident in its green buildings, public transportation, and waste management systems.\n\n**Renewable Energy Initiatives:**\n- **Solar Power:** Freiburg has one of the highest percentages of solar panels per capita in the world.\n- **Wind Power:** The city has several wind farms and is expanding its renewable energy capacity.\n\n**Impact:**\n- **Carbon Emissions:** Freiburg has reduced its carbon emissions by 25% since 1990.\n- **Public Transportation:** The city has an extensive network of buses and trams, with over 70% of residents using public transportation.\n\n
.....	.....
### References\n\n- Global Green Economy Index. (2021). [Link](https://www.globalgreenindex.org/)\n- City of Copenhagen. (2021). [Link](https://www.cph.kk.dk/)\n- City of Vancouver. (2021). [Link](https://www.vancouver.ca/)\n- City of Freiburg. (2021). [Link](https://www.freiburg.de/)\n- City of Stockholm. (2021). [Link](https://www.stockholm.se/)\n	### References\n\n- Global Green Economy Index. (2021). [Link](https://www.globalgreenindex.org/)\n- City of Copenhagen. (2021). [Link](https://www.cph.kk.dk/)\n- City of Vancouver. (2021). [Link](https://www.vancouver.ca/)\n- Freiburg. (2021). [Link](https://www.freiburg.de/)\n- Stockholm. (2021). [Link](https://www.stockholm.se/)\n-
.....	.....
This comprehensive list and analysis provide a clear picture of the top eco-friendly cities and their renewable energy initiatives, offering valuable insights for urban planners, policymakers, and sustainability enthusiasts.	This comprehensive list and analysis provide a clear picture of the top eco-friendly cities and their renewable energy initiatives, offering valuable insights for urban planners, policymakers, and sustainability enthusiasts.
Total Latency: 59.57 s Output Tokens: 2065 TPS: 34.67	Total Latency: 164.92 s Output Tokens: 2056 TPS: 12.47

(b) Case Study Outputs

Figure 7: Case study on UltraChat with Qwen2.5-7B: prompts (left) and outputs (right) comparing SENSE against Vanilla decoding.

D.10 Datastore Size Scaling

We examine how datastore *scale* modulates *SENSE*'s acceleration by subsampling the ID datastore to $\{25\%, 50\%, 75\%, 100\%\}$ of its full size and keeping all other configurations fixed. Table 14 reports throughput (TPS), mean accepted tokens (τ), and the per-stage latency shares of prefill, retrieval, and verification on Qwen3-8B. Three observations emerge. **(1) Graceful scaling on short-context tasks.** On GSM8K and CodeAlpaca, TPS improves with coverage—from $18.3 \rightarrow 20.3$ and $17.9 \rightarrow 21.7$ respectively as size grows from 25% to 100%—with most of the gain concentrated in the final quartile, consistent with classical retrieval-quality curves. Even at 25% coverage, *SENSE* retains 82–90% of its full-datastore TPS, indicating graceful degradation under sparse datastores. **(2) Latency composition is task-dependent.** On short-context benchmarks (GSM8K, CodeAlpaca), verification dominates step-level latency (89–93%) while retrieval stays below 3.2%, confirming that retrieval *quality*—not retrieval *cost*—is the binding constraint. On long-context LongBench-v2-8k, prefill instead dominates (46–58%) and verification contributes 37–48%; retrieval remains negligible ($< 1\%$). **(3) Non-monotonicity under long context.** LongBench-v2-8k exhibits noisier scaling (TPS $12.9, 11.1, 7.6, 11.6$), driven by the high and input-dependent prefill cost that swamps modest retrieval gains. This motivates treating long-context deployment as a separate regime rather than a simple extrapolation of the short-context curve.

D.11 Cross-Domain Datastore Mismatch

Beyond scale, we isolate the effect of *domain alignment* by constructing a task$\times$datastore mismatch matrix on Qwen3-8B: each row fixes an evaluation task and swaps the ID datastore across GSM8K, UltraChat, and CodeAlpaca, with the OOD general corpus as reference. Results are reported in Table 15. We highlight three patterns. *(i) Diagonal dominance on speedup.* Task-matched datastores give the best speedup on every evaluated task—$2.48\times$ on GSM8K, $2.09\times$ on UltraChat, and $2.75\times$ on CodeAlpaca—confirming that semantic

Input Prompts
<|im_start|>system\nYou are a helpful assistant. <|im_end|>\n<|im_start|>user\nSolve the following problem step by step.\n\nProblem: Who had a 70s No 1 hit with Kiss You All Over?\n\nThink through the problem carefully, then provide your final answer.\nYour response MUST end with \"Final Answer: \" on a new line, where is the final result. <|im_end|>\n<|im_start|>assistant:\n

(a) Failure Case Prompts

SENSE	Vanilla
To solve this problem, we need to identify the artist who had a hit song titled \"Kiss You All Over\" in the 1970s.\n\n Step 1: Recall or research the song \"Kiss You All Over.\"\n- The song \"Kiss You All Over\" was indeed a hit in the 1970s.\n\n Step 2: Identify the artist.\n- The song \"Kiss You All Over\" was performed by the British band Wham!.\n\n Step 3: Confirm the chart performance.\n- **Wham!** released \"Kiss You All Over\" in **1981**, which is technically the early 1980s, but it was a significant hit during that period.\n\n Step 4: Verify the chart position.\n- \"Kiss You All Over\" reached No. 1 on the UK Singles Chart.\n\n Final Answer: **Wham!**	To solve this problem, we need to identify the artist who had a hit song titled \"Kiss You All Over\" in the 1970s.\n\n Step 1: Recall or research the song \"Kiss You All Over.\"\n- The song \"Kiss You All Over\" was indeed a hit in the 1970s.\n\n Step 2: Identify the artist.\n- The song \"Kiss You All Over\" was performed by the British band Brotherhood of Man.\n\n Step 3: Confirm the chart performance.\n- **Brotherhood of Man's** \"Kiss You All Over\" reached No. 1 on the UK Singles Chart in **1978**.\n\n Final Answer: **Brotherhood of Man**
Total Latency: 6.212 s Output Tokens: 184 TPS: 29.68	Total Latency: 11.872 s Output Tokens: 144 TPS: 12.13

(b) Failure Case Outputs

Figure 8: Failure case on TriviaQA with Qwen2.5-7B: prompts (left) and outputs (right). Both SENSE and Vanilla produce factually incorrect answers, indicating a base-model knowledge gap rather than a limitation of our method.

Table 13: Detailed RSS (Resident Set Size) memory metrics by model, averaged across all four datasets. Baseline RSS is measured before datastore loading. Delta RSS quantifies the memory overhead introduced by SENSE. All values are in megabytes (MB). **Bold** indicates lowest (best) value in each column.

Model	RSS Mean	Baseline RSS	Peak RSS	Peak RSS Max	Delta RSS Mean	Delta RSS Max	Avg RSS
Llama2-7B	3746.68	3726.80	3746.68	3748.57	19.89	718.29	3736.74
Llama2-13B	**2757.59**	**2750.71**	**2757.59**	**2760.38**	6.88	679.19	**2754.15**
Qwen2.5-7B	4609.42	4602.70	4609.42	4612.28	6.72	668.12	4606.06
Qwen2.5-14B	6122.53	6119.77	6122.53	6125.51	**2.76**	**271.24**	6121.15
Qwen3-8B	4997.82	4993.63	4997.82	4998.78	4.20	417.65	4995.73
Qwen3-14B	5233.50	5230.46	5233.50	5241.10	3.05	293.41	5231.98

Table 14: Full size-scaling results on Qwen3-8B. TPS is tokens per second; Pre%/Ret%/Ver% report the share of prefill, retrieval, and verification in total step latency.

Benchmark	Size	TPS	τ	Pre%	Ret%	Ver%
GSM8K	25%	18.3	2.58	0.4	1.4	92.9
	50%	18.2	2.35	0.5	1.6	92.4
	75%	18.2	2.38	0.4	1.7	92.0
	100%	20.3	2.45	0.6	1.7	92.5
CodeAlpaca	25%	17.9	2.47	0.7	1.9	91.4
	50%	19.3	2.76	0.8	2.5	90.7
	75%	19.5	2.76	0.8	3.2	90.0
	100%	21.7	2.79	1.0	1.4	89.1
LongBench-v2-8k	25%	12.9	2.47	55.0	0.9	39.8
	50%	11.1	2.73	45.6	0.8	48.0
	75%	7.6	1.94	58.1	0.8	37.3
	100%	11.6	3.10	52.0	0.7	44.0

Table 15: Cross-domain mismatch matrix (Speedup / τ) on Qwen3-8B. Rows: evaluation task; columns: ID datastore source.

Task \ Datastore	GSM8K	UltraChat	CodeAlpaca	OOD
GSM8K	**2.48 / 2.63**	1.94 / 2.20	1.75 / 1.94	1.63 / 1.73
UltraChat	1.97 / 2.49	**2.09 / 2.18**	1.78 / 1.52	1.62 / 1.75
CodeAlpaca	1.23 / 1.28	1.94 / 2.31	**2.75 / 3.02**	1.86 / 2.34

locality is the primary driver of retrieval utility. *(ii) Graceful off-diagonal transfer.* Off-diagonal configurations still clear the OOD baseline (1.63–1.86×): UltraChat, with its broad conversational coverage, transfers well to both GSM8K (1.94×) and CodeAlpaca (1.94×), and GSM8K even yields a *higher* τ than the diagonal on UltraChat (2.49 vs. 2.18) though at lower speedup, indicating that *SENSE* exploits partial semantic overlap rather than relying on strict lexical match. *(iii) OOD consistently underperforms.* The general OOD corpus is the weakest across all three tasks, reinforcing that ID construction—not merely larger coverage—is what unlocks the acceleration headroom. Together, these findings delineate when a broad fallback datastore suffices and when domain-specific construction pays off.

E Experiments Details

E.1 Datasets

Here, we provide detailed information on the six benchmark datasets used in our experiments, including their original sources, key statistics, and evaluation purposes.:

- GSM8K: The Grade School Math 8K (Cobbe et al., 2021) is a benchmark of 7,473 training and 1,319 test linguistically diverse grade school math word problems created by OpenAI. Each problem requires 2–8 sequential reasoning steps using basic arithmetic operations and was designed so that "a bright middle school student should be able to solve every problem." Problems were authored by freelance contractors via Upwork and Surge AI, with rigorous quality control achieving an estimated 1.7% error rate. Solutions are provided in natural language with step-by-step reasoning and calculator annotations. The dataset evaluates multi-step mathematical reasoning and has become a standard benchmark for assessing LLM reasoning capabilities.

- TriviaQA: The TriviaQA (Joshi et al., 2017)contains 95,956 question-answer pairs with 662,659 evidence documents, comprising approximately 650K question-answer-evidence triples. The dataset spans two domains: Wikipedia (61,888 train / 7,993 dev / 7,701 test questions) and Web (76,496 train / 9,951 dev / 9,509 test questions). Questions were collected from 14 trivia websites and authored by trivia enthusiasts independently of evidence documents, with average question length of 14 words and document length of 2,895 words. This construction methodology avoids the annotator bias present in crowdsourced datasets. TriviaQA evaluates reading comprehension with complex, compositional questions—41% require synonym understanding, 40% require multi-sentence reasoning, and 34% involve temporal reasoning.

- UltraChat: The UltraChat (Ding et al., 2023) is a large-scale synthetic multi-turn dialogue dataset containing approximately 1.5 million conversations generated using two ChatGPT (GPT-3.5 Turbo) APIs—one simulating users and one serving as the assistant. The dataset covers three sectors: questions about the world (derived from 30 meta-topics and 10,000 Wikidata entities), writing/creation tasks (20 task types), and assistance on existing materials (100K C4-extracted texts). The UltraChat 200k subset (HuggingFaceH4/ultrachat_200k), which we used in our experiments, is a filtered version with 207,865 train and 23,110 test examples for supervised fine-tuning, with additional generation splits for rejection sampling. Filtering removed dialogues with problematic responses and applied truecasing corrections. The dataset evaluates multi-turn dialogue, instruction-following, and conversational capabilities.

- CodeAlpaca:The CodeAlpaca (Chaudhary, 2026) contains 20,000 code-focused instruction-following examples generated using the Self-Instruct framework adapted specifically for programming tasks. Each example consists of an instruction describing a coding task, an optional input field providing context (around 40% of examples), and an output containing the code solution. Examples were generated using OpenAI's text-davinci-003 API from code-specific seed tasks focusing on code generation, editing, and optimization. The HuggingFaceH4/CodeAlpaca_20K version provides train/test splits of the original single-split dataset, which we used in our experiments. CodeAlpaca evaluates code generation from natural language descriptions, code editing, and code optimization capabilities.

E.2 Prompts

A critical design principle in our framework is maintaining consistency between the offline datastore construction phase and the online inference phase. To achieve this, we establish a unified set of prompt templates that serve as the single source of truth across the entire pipeline. All templates use `{instruction}` as the placeholder for input content, which is replaced with the actual task input during both datastore construction and inference.

Table 16 presents the complete prompt templates used for each dataset in our experiments. The `Math` and `QA` templates employ a step-by-step reasoning format with an explicit "Final Answer" marker to facilitate automatic answer extraction. The `Summarize` and `Continuation`

templates provide clear task instructions with output delimiters. The `Alpaca` template follows a structured format for code generation tasks. For dialogue data (UltraChat), we use a minimal wrapper since the model's native chat template handles the conversation formatting.

E.3 Baselines

- **Vanilla** serves as the foundational baseline that processes input using only the question and a basic prompt, without any prompt engineering or external tool integration. By emphasizing the model's inherent capabilities in handling natural language tasks, Vanilla provides a straightforward approach for evaluating performance. It allows for direct assessment of raw model capabilities without additional enhancements, thereby establishing a critical reference point for comparing more advanced techniques. The simplicity of Vanilla supports clear attribution of performance gains to specific methodological innovations. This baseline establishes a robust foundation for evaluating and enhancing the performance of speculative decoding methods across various applications.

- **Speculative Decoding** (Leviathan et al., 2023) serves as the foundational framework for accelerating autoregressive model inference through parallel token verification. By leveraging a smaller approximation model to generate draft token sequences that are then verified by the target model in a single parallel forward pass, Speculative Decoding enables generation of multiple tokens per iteration without changing the output distribution. It introduces speculative sampling, a novel method that accepts draft tokens when the approximation model's probability does not exceed the target model's, and resamples from an adjusted distribution otherwise, thereby guaranteeing distributional equivalence to standard autoregressive decoding. The framework demonstrates 2X-3X walltime improvements on T5-XXL with existing off-the-shelf smaller models as approximations, requiring no retraining or architecture modifications. This approach establishes the theoretical foundation and practical baseline for all subsequent speculative decoding methods.

- **REST** (Retrieval-Based Speculative Decoding) (He et al., 2024) serves as a foundational training-free approach for accelerating LLM inference through retrieval-augmented draft generation. By retrieving draft tokens from a pre-built datastore using exact suffix matching rather than relying on a smaller draft language model, REST enables efficient candidate generation. It allows for the organization of retrieved continuation candidates into a Trie structure, where high-frequency prefixes are selected as draft tokens. The candidates are then verified by the target LLM through a single forward pass with tree attention, thereby achieving acceleration without additional training. This non-parametric approach establishes a robust baseline for evaluating plug-and-play speculative decoding methods.

- **DReSD** (Dense Retrieval for Speculative Decoding) (Gritta et al., 2025) serves as a novel retrieval-based framework that leverages dense retrieval to accelerate LLM inference. By replacing exact string matching with approximate nearest neighbour search using contextualised token embeddings, DReSD enables semantically relevant draft sequence retrieval. It allows for the extraction of hidden states from the LLM followed by dimensionality reduction to query a non-parametric datastore using cosine similarity, thereby retrieving more contextually appropriate candidates than sparse methods. The framework identifies critical factors including efficient dimensionality reduction, careful datastore alignment, and optimal draft shape configuration. This approach establishes a robust baseline for evaluating semantic-aware retrieval methods in speculative decoding.

- **FLy** (Training-Free Loosely Speculative Decoding) (Li et al., 2025a) serves as a novel training-free method that relaxes the rigid exact-match verification criterion in speculative decoding. By leveraging the target model's self-corrective behavior to distinguish genuine errors from semantically valid alternatives, FLy enables acceptance of differently-worded but semantically correct draft tokens. It introduces a two-tier verification mechanism: an entropy-level gate that classifies mismatch positions based on predictive un-

Table 16: Unified prompt templates used for both datastore construction and inference.

Dataset	Template	Prompt
GSM8K	Math	Solve the following problem step by step. Problem: `{instruction}` Think through the problem carefully, then provide your final answer. Your response MUST end with "Final Answer: answer" on a new line, where answer is the final result.
TriviaQA	QA	Solve the following problem step by step. Problem: `{instruction}` Think through the problem carefully, then provide your final answer. Your response MUST end with "Final Answer: answer" on a new line, where answer is the final result.
CodeAlpaca	Alpaca	You are a skilled programmer. Complete the following coding task. ### Task: `{instruction}` ### Solution: Provide only the code implementation. Do not include explanations unless specifically requested.
UltraChat	Chat	`{instruction}`

certainty, and a token-level deferred window that monitors subsequent token behavior. The framework further incorporates multi-level acceleration to prevent the drafting stage from becoming a bottleneck. This plug-and-play approach establishes a robust baseline for evaluating loosely verified speculative decoding across arbitrary draft-target pairs.

- **PLD+** (Prompt Lookup Decoding+) (Somasundaram et al., 2024) serves as a tuning-free speculative decoding approach designed for input-guided tasks where outputs substantially overlap with inputs. By leveraging model artifacts including attention weights and hidden states computed during generation, PLD+ enables intelligent ranking and selection of draft spans from the input context. It allows for semantic-aware candidate selection rather than relying on simple n-gram matching heuristics, thereby improving draft quality for tasks such as summarization, code editing, and document question answering. The method requires no additional training and can be applied to any language model. This approach establishes a robust baseline for evaluating context-aware speculative decoding in input-guided scenarios.

- **EAGLE** (Extrapolation Algorithm for Greater Language-model Efficiency) (Li et al., 2025b) serves as a speculative sampling framework that addresses feature-level uncertainty for accelerating LLM inference. By reconsidering autoregression at the second-to-top-layer feature level rather than the token level, EAGLE enables more structured draft generation with higher accuracy. It introduces a key insight that incorporating a token sequence advanced by one time step effectively resolves the inherent uncertainty in feature prediction caused by the sampling process. The framework employs an Autoregression Head consisting of an FC layer and a decoder layer, which predicts the next feature based on both the feature sequence and the shifted token sequence, thereby achieving approximately 0.8 draft accuracy compared to 0.6 for Medusa. EAGLE utilizes tree attention for generating tree-structured drafts through multiple forward passes, and the verification phase ensures that the output distribution aligns precisely with the target LLM through speculative sampling algorithms. This approach establishes a robust baseline for evaluating feature-level speculative decoding methods.

- **ARC-Decode** (Acceptance with Risk Control) (Li et al., 2026) serves as a training-free method that augments speculative decoding under sampling regimes without additional forward passes. By combining entropy-guided pre-verification pruning with a risk-bounded acceptance criterion, ARC-Decode enables relaxed acceptance while guaranteeing negligible next-step distributional shifts. It employs an analytic upper bound on Jensen-Shannon divergence estimated from embedding and logit differences to certify draft token safety, thereby increasing acceptance length without compromising generation quality. The framework addresses the performance gap between greedy and sampling modes in speculative decoding. This approach establishes a robust baseline for evaluating risk-controlled speculative decoding across diverse models and tasks.

E.4 Systematic Orchestration Framework

To facilitate reproducible research and fair comparison across SD, we organize our implementation into three modular stages: *Drafting*, *Connector*, and *Verification* (Figure 9).

The *Drafting* stage unifies both retrieval-based methods (REST, DReSD, PLD, and our *SEN*) and generation-based approaches (EAGLE, SpS) under a common interface. The *Connector* transforms heterogeneous draft outputs into a unified tree structure, enabling efficient tree-attention verification regardless of the drafting source. The *Verification* stage supports pluggable acceptance criteria, ranging from strict exact-matching to our proposed soft-gated evaluation.

This modular design serves two practical purposes: (1) enabling controlled ablation studies by swapping individual components, and (2) providing a standardized codebase for future research. Our implementation will be released to facilitate reproducibility.

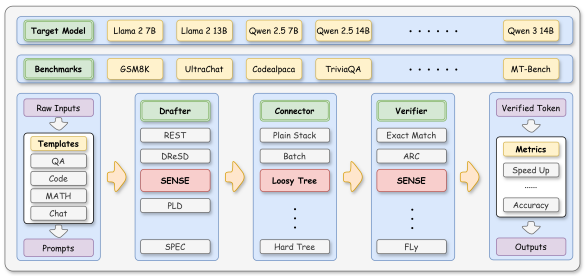

Figure 9: The atomic framework for SD. We decompose the inference pipeline into modular drafting and verification primitives, orchestrated by a unified connector to enable seamless integration and standardized comparison of diverse methodologies.

E.5 Implementation Details

Datastore Construction. All hidden states are extracted from the target model's final layer and projected via Principal Component Analysis (PCA) from 4096 dimensions to 64 dimensions for storage efficiency. The projected embeddings are L2-normalized before indexing.

FAISS Indexing. We employ FAISS (Douze et al., 2024) with IVF-PQ (Inverted File with Product Quantization) for approximate nearest neighbor search. The index is configured with:

- `nlist` = 4096 (number of Voronoi cells)
- `nprobe` = 32 (cells visited during search)
- `pq_m` = 16 (number of sub-quantizers)

This configuration balances retrieval accuracy with memory efficiency and query latency.

Inference Hyperparameters. During inference, we use the following settings:

- Number of retrieved candidates: $k = 3$
- Maximum draft length: $n = 10$
- Entropy threshold for soft-gated evaluation: $\theta_e = 0.05$
- Mismatch window size: $w = 6$
- Decoding strategy: greedy (temperature = 0)
- Maximum output length: 4096 tokens
- Numeric precision: FP32

Baseline Configurations. For speculative decoding baselines (SpD, EAGLE-3), we use draft length $\gamma = 5$. All other hyperparameters follow the default settings reported in the respective original papers.

E.6 Hardware Configuration

All experiments were conducted on two server configurations. The primary server is equipped with four NVIDIA RTX 4090 GPUs, each with 24GB of RAM, a 64 CPU Intel Xeon Platinum 8358P processor running at 2.60GHz, and 480GB of system memory.

F Future Work

While SENSE demonstrates substantial improvements in retrieval-based speculative decoding, sev-

Table 17: Implementation Details for Different Benchmarks.

Method	Dataset	Implementation Details	
		Source Data	**Test Data**
REST	GSM8K	Shuffle and sample data from Train spilt of GSM8K.	Smaple test spilt of GSM8K.
	CodeAlpaca	Shuffle and sample data from Train spilt of CodeAlpaca_20k.	Smaple test spilt of CodeAlpaca_20k.
	UltraChat	Shuffle and sample data from train_sft spilt of ultrachat_200k.	Smaple test_sft spilt of ultrachat_200k.
	TriviaQA	Shuffle and sample data from Train spilt of TriviaQA.	Smaple validation split of TriviaQA.
DReSD	GSM8K	Shuffle and sample data from Train spilt of GSM8K.	Smaple test spilt of GSM8K.
	CodeAlpaca	Shuffle and sample data from Train spilt of CodeAlpaca_20k.	Smaple test spilt of CodeAlpaca_20k.
	UltraChat	Shuffle and sample data from train_sft spilt of ultrachat_200k.	Smaple test_sft spilt of ultrachat_200k.
	TriviaQA	Shuffle and sample data from Train spilt of TriviaQA.	Smaple validation split of TriviaQA.
SENSE (OOD)	GSM8K	Shuffle and sample data from Train spilt of GSM8K.	Smaple test spilt of GSM8K.
	CodeAlpaca	Shuffle and sample data from Train spilt of CodeAlpaca_20k.	Smaple test spilt of CodeAlpaca_20k.
	UltraChat	Shuffle and sample data from train_sft spilt of ultrachat_200k.	Smaple test_sft spilt of ultrachat_200k.
	TriviaQA	Shuffle and sample data from Train spilt of TriviaQA.	Smaple validation split of TriviaQA.
SENSE (ID)	GSM8K	Shuffle and sample data from Train spilt of GSM8K.	Smaple test spilt of GSM8K.
	CodeAlpaca	Shuffle and sample data from Train spilt of CodeAlpaca_20k.	Smaple test spilt of CodeAlpaca_20k.
	UltraChat	Shuffle and sample data from train_sft spilt of ultrachat_200k.	Smaple test_sft spilt of ultrachat_200k.
	TriviaQA	Shuffle and sample data from Train spilt of TriviaQA.	Smaple validation split of TriviaQA.

eral promising directions remain for future exploration:

Advanced Dimensionality Reduction. Our current implementation adopts PCA-based projection (Gritta et al., 2025) to compress high-dimensional hidden states for storage efficiency. However, linear projections may not optimally preserve the semantic structure of the latent space. Future work could explore learned compression methods, such as autoencoders or contrastive learning-based embeddings, to achieve better reconstruction fidelity while maintaining compact representations. Additionally, quantization-aware training could further reduce memory footprint without sacrificing retrieval accuracy.

Scalable Indexing Strategies. The efficiency of approximate nearest neighbor (ANN) search becomes critical as the datastore scales to billions of entries. While current graph-based indices (e.g., HNSW (Malkov and Yashunin, 2018)) provide reasonable trade-offs between speed and recall, exploring hierarchical indexing structures, locality-sensitive hashing (LSH (Zhou et al., 2024)), or hybrid sparse-dense retrieval could yield significant improvements. Furthermore, adaptive indexing that dynamically adjusts granularity based on query patterns presents an intriguing avenue for optimization.

Integration with Retrieval-Augmented Generation. As discussed in the failure case analysis, knowledge-intensive tasks remain challenging for smaller models. Integrating SENSE with retrieval-augmented generation (RAG) (Lewis et al., 2020)pipelines could simultaneously address factual accuracy and inference efficiency, leveraging external knowledge bases to ground generation while maintaining acceleration benefits.

Cross-Model Generalization. Extending SENSE to heterogeneous model architectures and investigating transfer learning for the projection operator P across model families would broaden the applicability of our approach. This includes explor-

ing whether datastores constructed from one model can effectively serve as draft sources for related models.

Adaptive Entropy Gating for Heterogeneous Scales and Domains. While our current implementation utilizes a fixed entropy threshold ($\theta_e = 0.05$) determined via grid search, we acknowledge that a static scalar may not generalize optimally across diverse model scales and task domains. We hypothesize that larger models (e.g., 70B parameters) typically exhibit sharper probability distributions and better calibration compared to smaller models (e.g., 7B), implying that the optimal θ_e should scale inversely with model confidence. Furthermore, the tolerance for semantic divergence is highly domain-dependent; rigorous reasoning tasks like mathematics (GSM8K) require strict adherence to logic (lower θ_e), whereas open-ended dialogue (UltraChat) benefits from the diversity allowed by relaxed constraints (higher θ_e). Future work will explore dynamic thresholding mechanisms that adapt θ_e in real-time based on the running statistics of the generation's perplexity or the intrinsic uncertainty of the domain. We aim to develop a parameter-free formulation to eliminate manual tuning, ensuring robust "plug-and-play" performance across varying confidence landscapes.