Lossless Acceleration of Large Language Models with Hierarchical Drafting based on Temporal Locality in Speculative Decoding

Anonymous ACL submission

Abstract

Accelerating inference in Large Language Models (LLMs) is critical for real-time interactions, as they have been widely incorporated into real-world services. Speculative decoding, a fully algorithmic solution, has gained attention for improving inference speed by drafting and verifying tokens, thereby generating multiple tokens in a single forward pass. However, current drafting strategies usually require significant fine-tuning or have inconsistent performance across tasks. To address these challenges, we propose Hierarchy Drafting (HD), a novel lossless drafting approach that organizes various token sources into multiple databases in a hierarchical framework based on temporal locality. In the drafting step, HD sequentially accesses multiple databases to obtain draft tokens from the highest to the lowest locality, ensuring consistent acceleration across diverse tasks and minimizing drafting latency. Our experiments on Spec-Bench using LLMs with 7B and 13B parameters demonstrate that HD outperforms existing lossless drafting methods, achieving robust inference speedups across model sizes, tasks, and temperatures. Our code is anonymously released during the review process[1].

1 Introduction

With the growing demand for accelerating Large Language Model (LLM) inference to enable efficient real-time human-LLM interactions, Speculative Decoding (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023) has gained attention for providing a fully algorithmic solution with minimal drawbacks. While autoregressive decoding generates token by token, the decoding step in this method is divided into two sub-steps: *drafting*, where likely tokens are sampled externally from a lesser complex model, and *verifying*, where the sampled tokens are accepted or rejected by comparing with the LLM's actual output. By allowing the LLM to generate multiple tokens accepted in the verification phase, speculative decoding improves both the throughput and latency of LLM inference. Crucially, the efficiency of this approach depends on how draft tokens are generated, as performance gains hinge on the acceptance rate of these tokens (Chen et al., 2023). Therefore, subsequent approaches to speculative decoding have focused on developing drafting strategies that sample tokens closely aligned with the target model.

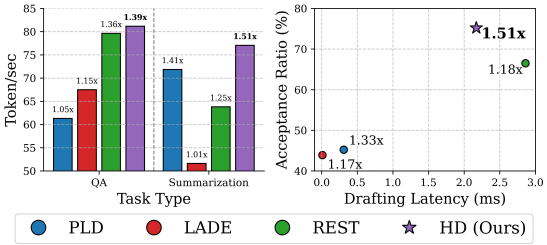

Figure 1: Result of database drafting methods on Spec-Bench (Xia et al., 2024) with Vicuna-7B (Zheng et al., 2023). The values in the plot denote the speedup against autoregressive decoding. (Left) QA and summarization task performance. (Right) Acceptance ratio and drafting latency.

Recent efforts in speculative decoding have focused on developing effective drafting methods, leveraging LM-based approaches, such as using smaller models than LLM (Zhou et al., 2024; Miao et al., 2024) or incorporating specialized branches within the LLM architecture (Cai et al., 2024; Li et al., 2024a). However, their applicability in real-world scenarios is limited by the significant overhead associated with fine-tuning for optimization. First, smaller models for drafting must be fine-tuned, such as through distillation, to generate tokens similar to LLMs for achieving optimal performance regardless of the given tasks (Zhou et al., 2024; Yi et al., 2024). In addition, current LLM families (Touvron et al., 2023; Zheng et al., 2023) do not offer models of an appropriate size for drafting, often necessitating training from scratch. In branch-based drafting, which modifies its original LLM architecture, the computational cost for training such branches within LLMs is significant due to

[1] https://anonymous.4open.science/r/Hierarchy_Drafting-F3F4

gradient calculations across the entire model, even though most parameters remain frozen (Cai et al., 2024; Li et al., 2024a,b). For example, EAGLE (Li et al., 2024b), one of the leading methods, needs 1-2 days of training on 2-4 billion tokens using 4 A100 GPUs for training 70B model.

To address these limitations, this paper explores a lightweight, lossless drafting strategy: *Database Drafting*, eliminating the need for parameter updates (Saxena, 2023; Fu et al., 2024; He et al., 2024). Database drafting constructs databases from various token sources and fetches draft tokens from the database using previous tokens. However, as previous work relies on a single database from a single source, the coverage of draft tokens is restricted, leading to inconsistent acceleration across different tasks, as depicted in the left side of Figure 1. For example, PLD (Saxena, 2023), which uses previous tokens as its source, shows strengths in the summarization, highly repeating the tokens in the earlier texts, yet it achieves only marginal speedups in QA, where fewer promising tokens are included in the prior text. A straightforward solution to improve coverage is incorporating diverse sources into a single database. However, increasing the database scale leads to higher drafting latency, resulting in additional overhead. As shown in the right side of Figure 1, REST (He et al., 2024), which uses the largest database, accurately predicts future tokens but suffers from significant latency, negating its high acceptance ratio benefits. Therefore, this paper proposes a solution to these limitations: *Utilize diverse token sources simultaneously for robust performance and minimal overhead.*

With this objective in mind, we propose a simple yet effective solution: **Hierarchy Drafting** (HD), which integrates diverse token sources into a hierarchical framework. Our proposed method is inspired by the memory hierarchy system, which prioritizes data with high *temporal locality* in the memory access for performance optimization (Aggarwal et al., 1987). Therefore, HD groups draft tokens from diverse sources based on their temporal locality—the tendency for some tokens to reappear within or across generation processes. For example, when an LLM solves a math problem like, '*The vertices of a triangle are at points (0, 0), (-1, 1), and (3, 3). What is the area of the triangle?*', the coordinates frequently repeat within a specific generation process but not across processes. In a related sense, phrases commonly generated by LLMs, such as '*as an AI assistant*', or frequent grammatical patterns exhibit relatively moderate locality, often appearing across different generation processes.

Based on their temporal locality, the multiple databases of HD organize them into *context-dependent database*, which stores tokens with high temporal locality for a given context; *model-dependent database*, which captures frequently repeated phrases by LLMs across generations; and *statistics-dependent database*, which contains statistically common phrases with slightly lower locality across processes than those in the model-dependent database. During inference, HD accesses the databases in order of temporal locality, prioritizing tokens with high locality by starting with context-dependent, then model-dependent, and finally statistics-dependent databases until a sufficient number of draft tokens are obtained to convey to the LLM for verification.

This strategy has two benefits: firstly, increasing drafting accuracy by leveraging temporal locality and secondly, reducing the overhead from drafting latency, as the scale of the databases is inversely correlated with the degree of locality—tokens with high locality are rarer. Thus, starting with the smaller context-dependent database for drafting tokens is more accurate and faster than using the larger statistics-dependent database alone. Also, our hierarchical framework can encompass other database drafting methods owing to its *plug-and-play* nature, making it easy to integrate diverse drafting sources based on their temporal locality.

We evaluate HD and other database drafting methods using widely adopted LLMs, Llama-2 (Touvron et al., 2023) and Vicuna (Zheng et al., 2023), on Spec-Bench (Xia et al., 2024), a benchmark designed to assess effectiveness across diverse tasks. Our proposed method, HD, outperforms other methods in our experiment and consistently achieves significant inference speedup across various settings, including model size, temperature, and tasks. We also analyze how the hierarchical framework adaptively selects the appropriate database for each task while minimizing draft latency, aligning with our design goals.

Our contributions in this paper are threefold:

- We identify the limitations of existing speculative decoding methods, which require additional fine-tuning or deliver inconsistent acceleration gains.
- We introduce a novel database drafting method, Hierarchy Drafting (HD), incorporating diverse

token sources into the hierarchical framework for robust performance with minimizing overhead.

- We demonstrate that HD consistently achieves significant acceleration gains across various scenarios compared to other lossless methods.

2 Related Work

We now introduce speculative decoding and lossless drafting strategies based on the database.

Speculative Decoding. Speculative decoding is a novel approach that accelerates LLM inference by minimizing the number of forward passes required, thereby reducing total latency (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). The core concept is that tokens, such as frequent phrases, can be predicted with high confidence using simpler models, enabling the generation of multiple tokens at once. Stern et al. (2018) introduced the *Draft-then-Verify* paradigm, dividing each decoding step into two sub-steps: drafting multiple tokens from draft models and verifying them against LLM outputs in parallel. This concept has been expanded by leveraging independent draft models and enabling various sampling methods (Xia et al., 2023; Leviathan et al., 2023; Chen et al., 2023). However, subsequent work emphasized aligning draft models closely with the target LLM for significant and consistent acceleration gains (Zhou et al., 2024; Miao et al., 2024; Yi et al., 2024). The typical solution is model distillation, which requires additional fine-tuning. Alternatively, additional branches can be incorporated into the target model architecture for better alignment, which still necessitates fine-tuning (Cai et al., 2024; Li et al., 2024b,a; Ankner et al., 2024). As a result, most speculative decoding methods involve extra computational costs, making actual lossless acceleration challenging to achieve.

Database Drafting. Database drafting eliminates training costs by retrieving draft tokens for previous inputs from a database rather than relying on smaller LMs or additional architectural branches. The database stores token pairs, with prefix tokens as keys and subsequent tokens as values. The sources of these databases vary across different methods, with each method relying on its own unique database source. Some approaches utilize input prompt tokens as draft sources, which is particularly effective for tasks like summarization or retrieval-augmented generation, where input tokens are frequently repeated during generation (Saxena, 2023; Yang et al., 2023). Another method retrieves draft tokens from large text corpora by leveraging language patterns (He et al., 2024). Although retrieval from large corpora introduces some latency overhead, the acceleration gained from accurate drafting typically outweighs this, resulting in faster inference overall. Additionally, LLMs can serve as sources for database drafting by generating tokens stored in the database, either through parallel decoding (Santilli et al., 2023; Fu et al., 2024) or token recycling (Luo et al., 2024), where tokens are relevant to the current generation process. Finally, the previously generated texts by LLMs can be served as draft token sources because LLMs frequently reuse specific phrases or words (Spector and Ré, 2023). Each source offers distinct strengths in predicting future tokens in certain scenarios, yet these strengths can become weaknesses in others. Therefore, it is worth noting that reliance on a single source may lead to limitations.

Table 1: Details of current database drafting methods with Vicuna-7B on Spec-Bench. Database scale measures the number of draft token sequences in each database. Drafting latency measures the average latency for the drafting step.

Method	Database Scale	Drafting Latency	#Token/Sec
PLD (Saxena, 2023)	519 ± 423	0.31 ± 0.06 ms	74.51 ± 23.09
LADE (Fu et al., 2024)	354 ± 258	< 0.01 ms	66.34 ± 14.09
REST (He et al., 2024)	200M	2.86 ± 6.43 ms	66.04 ± 14.72

Table 1 shows the experimental results of current database drafting methods, which construct their databases from a single source. Specifically, PLD (Saxena, 2023) exhibits the highest speedup compared to other approaches but also shows a significant standard deviation in speedup gains. This variability is attributed to the limited and uneven sizes of the databases, leading to inconsistent acceleration across the generation process. In contrast, LADE (Fu et al., 2024) achieves an impressively low drafting latency—less than 0.01 ms. However, this remarkable value does not translate to significant acceleration due to its small database size, akin to PLD. However, increasing database size alone, as demonstrated by REST (He et al., 2024), does not provide a viable solution for improving the effectiveness of database drafting. While a larger database scale can improve the accuracy of the drafting step, it also leads to higher latency since retrieving tokens from a larger database introduces additional processing overhead.

Therefore, to address the limitations of current lossless drafting methods relying on a single source, we propose integrating diverse sources into a hierarchical framework, aiming to harness each source's strengths more effectively with minimal overhead.

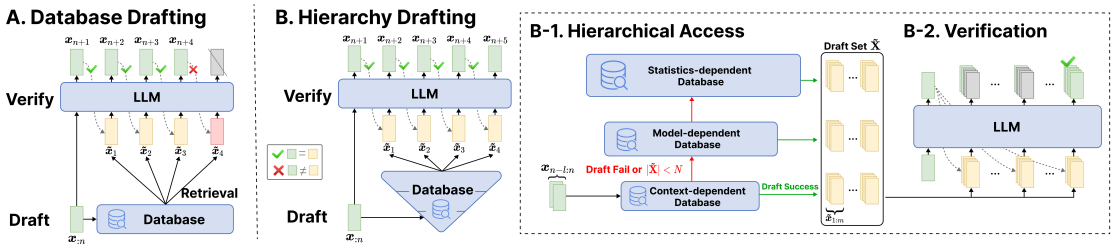

Figure 2: Overview of database drafting and our proposed method, Hierarchy Drafting (HD). *A.* Previous database drafting methods retrieve draft tokens from a single database constructed from a single source, leading to inconsistent acceleration gains across different scenarios. *B.* HD, however, leverages multiple databases encompassing diverse sources to improve token coverage, ensuring consistent performance. *B-1.* During the drafting process, databases are accessed sequentially from the smallest to the largest, based on the temporal locality of the token sequences. *B-2.* Multiple draft token sequences are verified in parallel, and the sequence with the highest number of accepted tokens is finally selected as the generated output.

3 Method

We begin by formally defining speculative decoding and database drafting and present our proposed method, Hierarchy Drafting (HD), which addresses the limitations of database drafting methods.

3.1 Preliminary

Speculative Decoding. At each step of speculative decoding, multiple tokens $\tilde{\boldsymbol{x}}_{1:m}$ (i.e., draft token sequence) are drafted from an approximate model $\mathcal{M}_q$ to predict future tokens of LLM $\mathcal{M}_p$ (i.e., target model) for previous text tokens $\boldsymbol{x}_{\leq t}$:

$$\tilde{\boldsymbol{x}}_{1:m} \sim_m \mathcal{M}_q(\boldsymbol{x}_{\leq t}). \tag{1}$$

All draft token sequence $\tilde{\boldsymbol{x}}_{1:m}$ are verified against the actual output of $\mathcal{M}_p$. For example, in the greedy decoding, the tokens $\boldsymbol{x}'_{t+1:t+m}$ are obtained for a given $\tilde{\boldsymbol{x}}_{1:m}$ and $\boldsymbol{x}_{\leq t}$ by solving the following equations in parallel:

$$\begin{cases} x'_{t+1} &= \arg\max P_{\mathcal{M}_p}(x|\boldsymbol{x}_{\leq t}), \\ x'_{t+2} &= \arg\max P_{\mathcal{M}_p}(x|\tilde{x}_1, \boldsymbol{x}_{\leq t}), \\ & \dots \\ x'_{t+m} &= \arg\max P_{\mathcal{M}_p}(x|\tilde{\boldsymbol{x}}_{1:m}, \boldsymbol{x}_{\leq t}). \end{cases} \tag{2}$$

Each token x'_{t+i} is verified against the corresponding draft token $\tilde{x}_{t+i}$, starting from $i = 0$ until the verification fails or $i = m$ is reached. To enhance the likelihood of acceptance, multiple draft token sequences $\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{x}}^i\}_{i=1}^N$ (i.e., draft set) are verified in parallel. The specialized attention mask implements the parallel verification of the draft set, not causal attention mask (Fu et al., 2024; Miao et al., 2024). In the sampling strategy, speculative sampling (Chen et al., 2023) is commonly used to accept more tokens while maintaining identical output distributions of the target model. In summary, the generation step is divided into two sub-steps with a single forward pass of the target model. The multiple accepted tokens are generated simultaneously, compressing the overall decoding process.

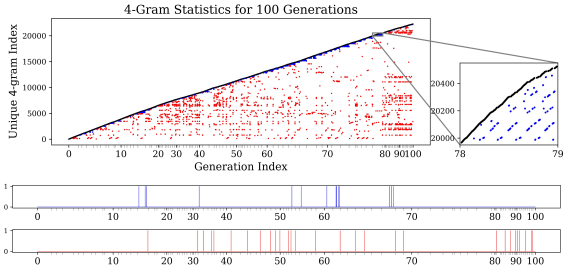

Figure 3: (Upper) 4-gram statistics for 100 generations of Llama-2-7b. The x-axis shows the order of 4-grams across 100 generations, with major ticks marking generation steps. The y-axis represents unique 4-gram indices. Red dots indicate 4-grams from previous processes, while blue dots represent those from the current process. (Lower) Frequency analysis of two 4-grams, represented by red and blue dots, respectively.

Database Drafting. As shown on the left side of Figure 2, the methods included in database drafting exploit the database $\mathcal{D}$, having the prefix tokens as the key and the subsequent tokens as the value. Per each step of the generation process, the draft token sequence $\tilde{\boldsymbol{x}}_{1:m}$ is retrieved from database $\mathcal{D}$ for given previous tokens $\boldsymbol{x}_{t-l:t}$:

$$\tilde{\boldsymbol{x}}_{1:m} \in \tilde{\boldsymbol{X}} = \texttt{Ret}(\boldsymbol{x}_{t-l:t}; \mathcal{D}), \tag{3}$$

where l and m are the length of previous tokens and draft token sequence. Subsequently, the verifying step is the same as other methods.

3.2 Hierarchy Drafting

We introduce Hierarchy Drafting (HD), which organizes tokens from diverse sources into three databases based on temporal locality and accesses them in order from the smallest to the largest scale. The overview and decoding process are depicted on the right side of Figure 2.

Observation: Temporal Locality. The main idea behind database drafting is that some tokens are easy to retrieve from the database because they exhibit temporal locality—meaning they tend to be repeated within or across the generation processes. However, note that not all draft token se-

quences share the same level of temporal locality during generation. We analyze the pattern of unique 4-grams during 100 text generations on Spec-Bench (Xia et al., 2024), as shown in Figure 3. The results reveal that certain 4-grams are frequently repeated and exhibit varying locality levels. Specifically, the blue dots and the right small plot in Figure 3 illustrate local redundancy, where the same 4-gram appears multiple times within a single generation step. This reflects high temporal locality within a single generation rather than across multiple generations. In contrast, the red dots in Figure 3 highlight a pattern where the model repeatedly generates the same 4-grams at different stages of the generation process, illustrating its tendency to reuse familiar sequences over time. Additionally, the lower plot of Figure 3 presents the frequency study of sampled red and blue dots, demonstrating that some tokens exhibit high temporal locality within a specific context, while others maintain consistent locality across generation processes. Therefore, given the varying temporal locality of tokens throughout the generation process, drafting steps should prioritize tokens with higher temporal locality over others.

Database Design. Based on the temporal locality of draft token candidates, we design three types of databases to categorize them. **1) Context-dependent DB** ($\mathcal{D}_c$) contains tokens highly relevant to the specific context of the generation process, such as the blue dots in the Figure 3. This includes tokens from the input prompt, tokens generated through parallel decoding, tokens discarded during the generation process, and others that are highly relevant to a given context. $\mathcal{D}_c$ is lookup table with the prefix tokens, $\boldsymbol{x}_{1:l}$, as the key and the subsequent tokens, $\boldsymbol{x}_{l:l+m}$, as the value. Also, $\mathcal{D}_c$ is consistently updated during each forward step and initialized when the following generation process is started. The database follows the Least Recently Used (LRU) policy for draft sequence updates. **2) Model-dependent DB** ($\mathcal{D}_m$) stores tokens frequently generated by LLM regardless of context, as represented by the red dots in Figure 3. Top-k frequently generated token sequences, $\boldsymbol{x}_{1:l+m}$, are sampled from the model-generated texts, with $\boldsymbol{x}_{1:l}$ as the key and $\boldsymbol{x}_{l+1:l+m}$ as the value. For $\mathcal{D}_c$ and $\mathcal{D}_m$, the maximum size of values for a single key is the same as the maximum draft set size N. **3) Statistics-dependent DB** ($\mathcal{D}_s$) draws its tokens from large text corpora to capture

Algorithm 1 Decoding Process with Hierarchy Drafting

Require: Target LLM $\mathcal{M}_p$, databases $(\mathcal{D}_c, \mathcal{D}_m, \mathcal{D}_s)$, input text sequence $\boldsymbol{x}_{\leq t}$, target sequence length T, the size of prefix tokens l, the size of draft token sequence m, the size of draft set N;

$n \leftarrow t$
while $n < T$ and `[EOS]` $\notin \boldsymbol{x}_{1:n}$ **do**
  // Drafting Step: Hierarchical access to three databases until the size of the draft set $\tilde{\boldsymbol{X}}$ is N.
  $\tilde{\boldsymbol{X}} \leftarrow \texttt{Ret}(\boldsymbol{x}_{n-l:n}; \mathcal{D}_c)$
  if $|\tilde{\boldsymbol{X}}| < N$ **then**
    $\tilde{\boldsymbol{X}} \leftarrow \tilde{\boldsymbol{X}} \cup \texttt{Ret}(\boldsymbol{x}_{n-l:n}; \mathcal{D}_m)$
  end if
  if $|\tilde{\boldsymbol{X}}| < N$ **then**
    $\tilde{\boldsymbol{X}} \leftarrow \tilde{\boldsymbol{X}} \cup \texttt{Ret}(\boldsymbol{x}_{n-l:n}; \mathcal{D}_s)$
  end if
  // Verification Step: Verify the draft token sequence in $\tilde{\boldsymbol{X}}$ and generate additional tokens for updating $\mathcal{D}_c$.
  $\boldsymbol{x}_{n:n+i}, \hat{\boldsymbol{x}} \sim_i \mathcal{M}_p(\boldsymbol{x}_{\leq n}, \tilde{\boldsymbol{X}})$
  $\mathcal{D}_c \leftarrow \text{Update}(\mathcal{D}_c, \hat{\boldsymbol{x}})$
  $n \leftarrow n + i$
end while

universal phrases commonly used in the language. Although these tokens are frequent, they occur less consistently across processes than those in $\mathcal{D}_m$. To efficiently retrieve the sequence from a large corpus, we utilize a suffix array (Manber and Myers, 1993) following the implementation of He et al. (2024). Implementation details are in section 4.

Our database design yields three distinct advantages. First, it integrates diverse sources into multiple databases, enabling us to leverage each source's strengths for robust acceleration across various tasks. Then, each database's size decreases as the tokens' temporal locality increases since tokens with higher locality are rarer, providing an opportunity to optimize drafting latency. Finally, the design is *plug-and-play*, easily integrating additional token sources by assigning them to the appropriate database based on their temporal locality.

Hierarchical Access. Using the three databases designed with the temporal locality in mind, we retrieve draft token sequence $\tilde{\boldsymbol{x}}_{1:m}$ for the given previous input $\boldsymbol{x}_{t-l:t}$. Database access order is based on the degree of temporal locality within the current generation process; thereby, the access starts with $\mathcal{D}_c$. Access then proceeds to $\mathcal{D}_m$, which has high locality across generations, and finally $\mathcal{D}_s$, with moderate locality across generations, until draft set $\tilde{\boldsymbol{X}}$ accumulates a sufficient number of candidates as pre-defined hyperparameter N. These accesses leverage the locality of the draft token sequence to enhance drafting accuracy and minimize latency overhead, preserving the benefits of drafting.

Decoding Process. We introduce the inference

process of speculative decoding with our proposed method, HD. First, for a given previous input $\boldsymbol{x}_{t-l,t}$, we acquire the set of draft token $\tilde{\boldsymbol{X}}$ from the three databases with hierarchical access. Then, the target LLM $\mathcal{M}_p$ verifies the draft token sequences while simultaneously generating the additional tokens $\hat{\boldsymbol{x}}$. These tokens are used to update the context-dependent DB either through parallel decoding (Santilli et al., 2023; Fu et al., 2024) or by recycling wasted tokens (Luo et al., 2024). These processes are repeated iteratively until either the `[EOS]` token is generated or the sequence reaches the pre-defined maximum length T. Details of the decoding are depicted in Algorithm 1.

4 Experimental Setup

We introduce the details of the experiment setups conducted to evaluate the effectiveness of HD.

Dataset. We exploit Spec-Bench (Xia et al., 2024), a comprehensive benchmark to evaluate speculative decoding across various tasks. The benchmark includes six representative tasks: Multi-turn Conversation, Translation, Summarization, Question Answering (QA), Mathematical Reasoning, and Retrieval-Augmented Generation (RAG). Each task has 80 instances, making a total of 480 generations.

Model. We utilize two publicly available LLM families: **Llama-2-chat** (Touvron et al., 2023) and **Vicuna-v1.3** (Zheng et al., 2023). Additionally, 7B and 13B models are exploited to assess the effectiveness across different model scales.

Baseline Method. We compare our proposed method, **HD**, with autoregressive decoding and various database drafting methods to validate its effectiveness. Specifically, **1) Autoregressive decoding (AR)** serves as an indicator for measuring acceleration gains. We also include **2) PLD** (Saxena, 2023), utilizing previous input prompts as a database, **3) LADE** (Fu et al., 2024), employing parallel decoding via a Jacobian iteration method, and **4) REST** (He et al., 2024), which retrieves draft tokens from a large text corpus.

Evaluation Metric. We utilize a variety of metrics to evaluate drafting overhead, drafting accuracy, and acceleration gain. To measure drafting overhead, we use **1) Drafting Latency**, which refers to the time taken to fetch draft tokens. Following Zhou et al. (2024), the drafting accuracy is assessed using **2) Acceptance Ratio** (α) and **3) Mean Accepted Tokens** (τ). The acceptance ratio (α) represents the ratio of accepted tokens to total tokens, while the mean accepted tokens (τ) denotes the expected number of accepted tokens per step. Finally, acceleration gain is measured using **4) #tokens/sec**, the number of tokens generated per second, and the **5) Speedup Ratio**, which compares #tokens/sec of each method to the value from autoregressive decoding.

Implementation Detail. The proposed method, HD, is configured with the hyperparameters l, m, N, and T set to 2, 4, 7, and 1024, respectively. To adopt a sampling strategy, we exploit speculative sampling (Chen et al., 2023) by setting draft probability as 1.0. For the context-dependent database ($\mathcal{D}_c$), the previous input tokens and the tokens generated via parallel decoding are included. For parallel decoding, we follow the implementation proposed by LADE (Fu et al., 2024), which allows simultaneous processing of the parallel decoding and verification branches. Therefore, following the implementation of LADE, the verifying step is based on *n*-gram. For the model-dependent database ($\mathcal{D}_m$), we collect LLM-generated texts from the English portion of the OASST training set (Köpf et al., 2023), using a 7B model from the targeted LLM family. A total of 39,283 texts were generated, from which we sampled the 100k most frequent token sequences. Lastly, for the statistics-dependent database ($\mathcal{D}_s$), we adopt the setting of REST (He et al., 2024), utilizing data sourced from UltraChat (Ding et al., 2023), with the database size being approximately 12GB. More details of the experimental setup are in Appendix B.

5 Results

We now present the experimental results on Spec-Bench, along with an in-depth analysis of HD.

5.1 Main Result

Table 2 presents our main results, averaged across all cases of Spec-Bench using three models, at both low temperature ($T = 0.0$) and high temperature ($T = 1.0$). First, our proposed method, HD, achieves the outperforming acceleration gain across all scenarios. In detail, when temperature is 0.0, HD achieves over 1.5x faster inference speed compared to autoregressive decoding, whereas the other methods fail to exceed a 1.4x speedup. Also, while the acceleration gain at $T = 1.0$ is slightly lower than $T = 0.0$, HD still achieves the fastest inference speed compared to all other methods across all models. These results demonstrate that our hier-

Table 2: Results of Hierarchy Drafting and various database drafting methods on Spec-Bench. The best results are **bold**.

	Method	Vicuna-7B-v1.3					Vicuna-13B-v1.3					Llama-2-7B-chat				
		D.L (ms)	α (%)	τ	#tokens/s	Speedup	D.L (ms)	α (%)	τ	#tokens/s	Speedup	D.L (ms)	α (%)	τ	#tokens/s	Speedup
$T = 0.0$	**Autoregressive Decoding**	-	-	1.00	56.39	1.00x	-	-	1.00	34.48	1.00x	-	-	1.00	55.54	1.00x
	PLD (Saxena, 2023)	0.31	45.22	1.62	74.51	1.32x	0.31	44.49	1.57	46.80	1.36x	0.31	35.02	1.36	65.48	1.18x
	LADE (Fu et al., 2024)	**<0.01**	43.92	1.64	66.34	1.18x	**<0.01**	43.59	1.63	41.59	1.21x	**<0.01**	50.03	1.59	67.76	1.22x
	REST (He et al., 2024)	2.85	66.51	1.82	66.04	1.17x	3.12	66.80	1.82	47.69	1.38x	2.85	72.33	1.92	76.65	1.38x
	Hierarchy Drafting (HD; Ours)	2.17	**75.21**	**2.38**	**85.23**	**1.51x**	2.30	**75.05**	**2.30**	**54.42**	**1.58x**	2.18	**79.72**	**2.31**	**91.22**	**1.64x**
$T = 1.0$	**Autoregressive Decoding**	-	-	1.00	55.19	1.00x	-	-	1.00	34.00	1.00x	-	-	1.00	54.46	1.00x
	LADE (Fu et al., 2024)	**0.01**	38.09	1.49	61.37	1.11x	**0.01**	40.27	1.51	39.68	1.16x	**0.01**	48.71	1.55	65.22	1.20x
	REST (He et al., 2024)	2.79	66.91	1.84	64.58	1.17x	3.09	67.61	1.84	47.10	1.39x	2.81	73.00	1.93	75.46	1.39x
	Hierarchy Drafting (HD; Ours)	2.41	**69.11**	**2.06**	**73.14**	**1.32x**	2.40	**70.61**	**2.11**	**49.19**	**1.44x**	2.15	**78.88**	**2.25**	**84.18**	**1.54x**

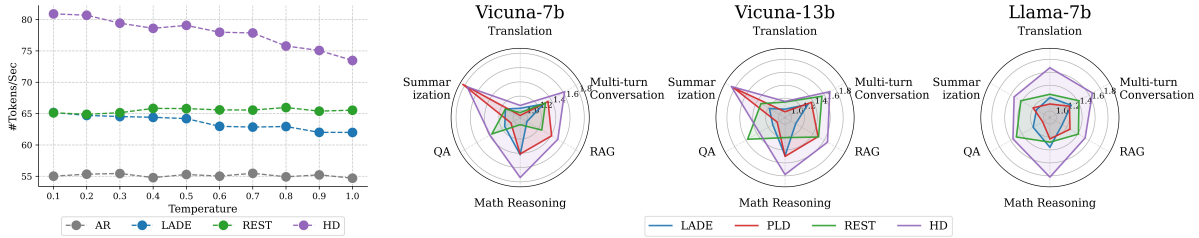

Figure 4: (Left) Tokens per second across varying temperature settings with Llama-2-7b for autoregressive decoding and database drafting methods. (Right) Speedup comparison of database drafting methods across six tasks of Spec-Bench.

archical framework effectively enhances inference speed by incorporating diverse token sources into three databases organized by temporal locality.

Beyond acceleration gains, we analyze the additional latency caused by the drafting process, which adds overhead that is absent in autoregressive decoding, and also evaluate how accurately the drafting step retrieves tokens that align with the LLM's output. Regarding drafting latency, LADE requires an extremely short time—under 0.01 ms per draft—whereas REST takes significantly longer, with latency close to 3.00 ms. However, drafting accuracy shows the opposite trend: LADE exhibits lower values for both the acceptance ratio (α) and mean accepted tokens (τ), while REST achieves higher values for both. Notably, our proposed method, HD, drafts slightly faster than REST, even though accessing the same extensive database, and accurately predicts 70% of generated tokens, achieving the highest accuracy among all other methods. These results indicate that HD successfully balances increased accuracy with reduced drafting latency through hierarchical database access, resulting in significant acceleration gain.

Impact of Temperature. As temperature sampling is commonly used to increase the diversity of text generation, we analyze its impact on the database drafting methods, as shown on the left of Figure 4. Other drafting methods maintain higher speeds than autoregressive decoding across all temperatures, with LADE slightly decreasing as temperature increases, whereas REST remains consistent. However, updating $\mathcal{D}_c$ with sampled outputs, similar to LADE, results in token mismatches at higher temperatures, causing HD to show a slight reduction in speed. Nevertheless, our proposed method, HD, outperforms all others across the entire temperature range, maintaining a high decoding speed of over 70 tokens per second. These results demonstrate that HD remains a suitable solution even with a sampling strategy, enhancing its potential in real-world scenarios.

Robustness across Tasks. We evaluate the robustness of database drafting methods across various generation tasks, as illustrated on the right side of Figure 4. Relying on a single source results in variability in acceleration gains, causing most methods, except HD, to show inconsistent performance with concave regions in specific tasks. Specifically, PLD achieves significant acceleration in tasks like Summarization and RAG but offers minimal improvements in Translation and QA. Additionally, other methods exhibit varying acceleration gains depending on the model used—REST, for example, performs well with Llama-7b in summarization but shows weaker results with Vicuna-7b, nearly matching autoregressive decoding speeds. In contrast, our proposed method consistently achieves the highest acceleration across all tasks and models, occupying the largest area in each plot. This demonstrates that incorporating diverse sources enhances robustness, making database drafting methods more suitable for real-world scenarios.

5.2 Analysis

In this subsection, we provide an in-depth analysis of HD for investigating its effectiveness.

Analysis of Three Databases. The left side of Fig-

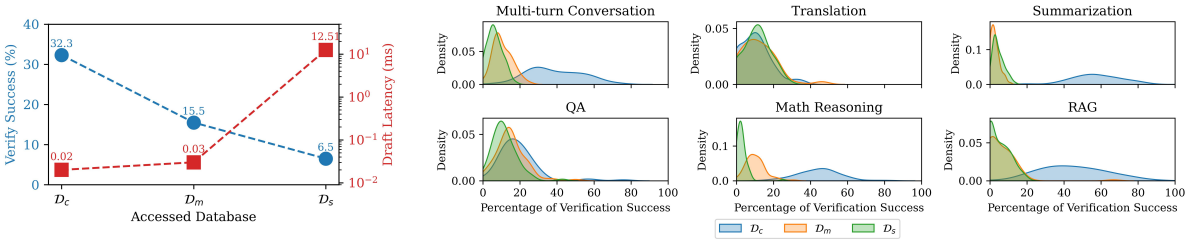

Figure 5: (Left) Verify success and draft latency for the databases $\mathcal{D}_c$, $\mathcal{D}_m$, and $\mathcal{D}_s$ in HD. Verify success represents the proportion of accepted accesses relative to the total accesses. (Right) Verify success density plots for each database across six tasks in Spec-Bench. Both results are conducted on Spec-Bench by using Llama-2-7b.

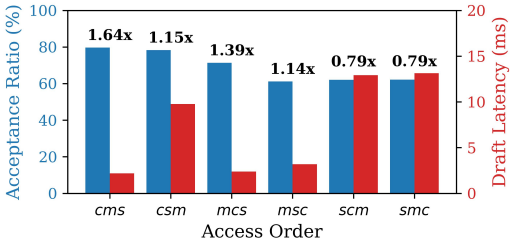

Figure 6: Impact of access order in HD with Llama-2-7b on Spec-Bench. Blue and red bars depict the acceptance ratio and draft latency, respectively. The value over the bars denotes the speedup against autoregressive decoding.

ure 5 depicts each database's verify success ratio and draft latency. The verify success ratio measures the proportion of accepted cases relative to total database accesses during the verifying step. $\mathcal{D}_c$ achieves the highest verify success ratio over 30% with the lowest draft latency, demonstrating its effectiveness in fetching context-relevant future tokens. However, $\mathcal{D}_m$ shows a lower verify success rate, 15.5%, with slightly higher latency, indicating that while it performs decently, it is less aligned with specific contexts. $\mathcal{D}_s$ exhibits the lowest verify success rate under 10% and the highest draft latency over 10ms due to its larger scale and lower locality. These highlight that draft tokens with higher locality are more frequently accepted, indicating alignment with our design objectives.

Access Pattern across Tasks. We analyze how our proposed method, HD, achieves consistent acceleration gain across tasks with verify success ratio of databases for each task. As shown in the right side of Figure 5, $\mathcal{D}_c$ excels in tasks such as Multi-turn Conversation or Summarization, where the context-specific tokens are highly repeated, leading to high verification success. However, for tasks like translation and QA, which offer fewer explicit cues from previous inputs or contexts, $\mathcal{D}_c$ achieves lower verification success. In these cases, $\mathcal{D}_m$ and $\mathcal{D}_s$ compensate for the weaknesses of $\mathcal{D}_c$ by showing higher verification success compared to other tasks where $\mathcal{D}_c$ outperforms. These results highlight how our HD efficiently accesses the appropriate database for each task, effectively leveraging the distinct strengths of diverse sources.

Database Access Order. We analyze the impact of access order within the hierarchical framework, as shown in Figure 6. As expected, our original access order (cms), which prioritizes databases from highest to lowest temporal locality, achieves the highest acceptance ratio and lowest draft latency. While other orders maintain an acceptance ratio above 50%, sufficient for some acceleration gain, their actual speedup is significantly lower due to additional drafting latency, reaching up to 12ms for orders like scm or smc. These results demonstrate that hierarchical access fully leverages the potential of multiple databases with minimal drafting latency compared to other orders, underscoring the importance of temporal locality in drafting order.

We provide additional analysis, including ablation study, token coverage of databases, database access statistics, and case study in Appendix C.

6 Conclusion

This paper explores the database drafting approaches in speculative decoding, which do not require additional training or fine-tuning. Existing methods rely on a single database from a single source, resulting in inconsistent and suboptimal acceleration gains. To address this, we propose Hierarchical Drafting (HD), which optimally utilizes diverse sources by constructing multiple databases based on temporal locality. Our method hierarchically accesses these databases, prioritizing those with the highest locality for optimal acceleration. Experimental results show that HD consistently and effectively accelerates LLM inference across various scenarios, outperforming other database drafting methods. These findings demonstrate that our hierarchical framework maximizes the strengths of each database with minimal overhead, expanding the directions exploiting multiple databases for lossless acceleration in speculative decoding.

Limitation

One limitation of this paper is the limited use of LLMs with more than 13B parameters. While our evaluation focused on models like Llama-2 and Vicuna with up to 13B parameters, the performance of HD on larger models remains unexplored. However, we expect that the larger models will be much more appropriate for our approach, considering the high acceptance ratio of our proposed method, HD, across diverse scenarios and decreased sensitivity to the draft latency as generation latency is increased. Also, we plan to extend our experiments to larger models in future work.

While this paper leverages multiple databases to maximize their strengths with minimal overhead, the sources of these databases are not entirely new. Rather than focusing on the novelty of each source, we emphasize that our approach is *plug-and-play*, making it easy to integrate future methods by simply adding tokens from new sources into the appropriate database. For instance, although we omitted token recycling (Luo et al., 2024) in our experiments, recycled tokens could be added to the context-dependent database, given their temporal locality.

Ethics Statements

This work proposes a lossless drafting strategy in speculative decoding for optimal and general acceleration gains. However, our method may generate violent or biased responses, which is beyond the scope of this paper. We strongly believe that future research on large language models will address these issues and help mitigate such concerns.

References

Alok Aggarwal, Bowen Alpern, Ashok K. Chandra, and Marc Snir. 1987. A model for hierarchical memory. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 305–314. ACM.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 12–58. The Association for Computer Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, abs/2302.01318.

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*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3029–3051. Association for Computational Linguistics.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. 2024. Ai and memory wall. *IEEE Micro*, 44(3):33–39.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2023. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:23056.07629*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Minh Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Alexandrovich Glushkov, Arnav Varma Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Julian Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Turning trash into treasure: Accelerating inference of large language models with token recycling. *arXiv preprint arXiv:2408.08696*.

Udi Manber and Eugene W. Myers. 1993. Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22(5):935–948.

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 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*, pages 932–949. ACM.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

David A. Patterson. 2004. Latency lags bandwith. *Commun. ACM*, 47(10):71–75.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. In *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12336–12355. Association for Computational Linguistics.

Apoorv Saxena. 2023. Prompt lookup decoding.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Benjamin Spector and Christopher Ré. 2023. Accelerating LLM inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10107–10116.

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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3909–3925. Association for Computational Linguistics.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 7655–7671. Association for Computational Linguistics.

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*, abs/2304.04487.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Euiin Yi, Taehyeon Kim, Hongseok Jeung, Du-Seong Chang, and Se-Young Yun. 2024. Towards fast multilingual LLM inference: Speculative decoding and specialized drafters. *arXiv preprint arXiv:2406.16758*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2024. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*.

Appendix

In the Appendix, we introduce more related work and details on how the experiments were implemented, along with supplementary results and analysis not included in the main text.

A More Related Work

LLM Inference Acceleration Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023) have significantly advanced Natural Language Processing (NLP) and are widely used in real-world applications via APIs, highlighting the importance of real-time human-LLM interactions. However, the slow inference speed of LLMs presents the primary bottleneck in deploying these models as practical services. This issue is predominantly memory-bound, with latency arising from the autoregressive decoding process (Patterson, 2004; Pope et al., 2023; Gholami et al., 2024). During the inference process, each token generation requires transferring model parameters and key-value caches from global memory to the accelerator's cache, consuming substantial overhead. Given the limited progress in memory bandwidth, several algorithmic approaches have been developed to mitigate these challenges, including model parameter quantization (Yao et al., 2022; Frantar et al., 2023; Kim et al., 2023), memory allocation optimization (Shazeer, 2019; Zhang et al., 2023; Kwon et al., 2023), and Speculative Decoding (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023). While other approaches often require hardware modifications or lead to side effects like performance degradation, speculative decoding has gained particular attention for offering a fully algorithmic solution with minimal drawbacks.

LM-based Drafting The straightforward approach for the drafting strategy of speculative decoding involves using an additional language model (LM) specialized for drafting (Leviathan et al., 2023; Chen et al., 2023; Miao et al., 2024; Zhou et al., 2024). To ensure effective drafting, such LMs must follow the target model's generation pattern and be smaller to minimize additional latency costs. LMs with parameter sizes under a billion are typically preferred for drafting, but currently, widely used LLM families do not usually have appropriate models. For example, the smallest officially available Llama-2 model (Touvron et al., 2023), with 7 billion parameters, is too large and inefficient for drafting purposes. Therefore, such methodologies often require training overhead to get the suitable LM for the targeted LLM, such as the distilled models from the target models (Miao et al., 2024) or lightweight models trained for mobile devices (Zhang et al., 2024).

Instead of using a separate language model for drafting, some approaches enhance the drafting capabilities of the target model itself (Cai et al., 2024; Li et al., 2024a,b; Ankner et al., 2024). In this line of work, the additional layer or branch in the target model is integrated into the target model to predict several subsequent tokens more than the very next token based on the last hidden states of given inputs. Following Stern et al. (2018), which exploits multiple heads for parallel decoding, Medusa (Cai et al., 2024) first integrates additional decoding heads into the target model. Subsequently, the branch-based drafting methodologies (Li et al., 2024a,b; Ankner et al., 2024) show remarkable effectiveness in sampling appropriate future tokens with achieving state-of-the-art results. However, integrating these layers or branches still requires significant training overhead. To sum up, branch-based drafting methods achieve remarkable speedup gains yet require additional computational costs, which are not trivial and are a new type of overhead for implementing speculative decoding.

B More Implementation Details

Hyperparameter First, l denotes the length of the previous tokens used as the database key, and m represents the length of the draft sequence used as the database value. N specifies the size of the draft set passed to the LLM for verification. Lastly, T is for the maximum length of generated tokens.

Implementation Details of HD First, we introduce more details of our multiple databases. For $\mathcal{D}_c$ and $\mathcal{D}_m$, the prefix token length used as the key is 1, while the given previous token length is l. This mismatch is because increasing the prefix length often leads to draft misses (i.e., keys not found in the database) due to the limited scale of token sources constructing $\mathcal{D}_c$ and $\mathcal{D}_m$. Also, for $\mathcal{D}_s$, the retrieval process is repeated by reducing the previous token length until draft token sequences are found or the token length reaches 0, following implementation of REST (He et al., 2024). As mentioned in the main text, $\mathcal{D}_c$ and $\mathcal{D}_m$ are lookup tables implemented by the Python Dictionary class.

In contrast, $\mathcal{D}_s$ is implemented by DraftRetriever[2], a Python Library based on suffix arrays, proposed by REST for handling a large text corpus with minimal overhead. The average number of values (i.e., draft token sequences) stored in the database is 1K, 100K, and 200M for $\mathcal{D}_c$, $\mathcal{D}_m$, and $\mathcal{D}_s$, respectively.

We now explain the detailed implementation of the decoding process with HD. Our method is primarily based on the implementation of LADE[3] (Fu et al., 2024), as LADE employs a similar database to $\mathcal{D}_c$ and updates tokens generated through parallel decoding. We extend this by incorporating a hierarchical framework; thereby, the verification step in HD follows LADE's n-gram verification process. However, we emphasize that HD can be extended to other database drafting methods by integrating a hierarchical drafting framework with multiple databases into their implementation.

Prompting Format We use the chat format for text input, provided by the Python library FastChat[4], which consists of system, user, and model components. Additionally, we adopted the system message for Vicuna from FastChat and for Llama-2 from the official repository[5].

Experimental Setup All experiments are conducted on a machine equipped with a single A100-40GB-PCIe GPU, using float16 precision for the models. To ensure a fair comparison, we follow the implementations of other database drafting methods and the evaluation scripts provided by Xia et al. (2024)[6]. Our experimental results are based on a single run, though we observed only marginal differences between runs.

Details of Dataset We exploit Spec-Bench (Xia et al., 2024), collecting data from representative datasets for each task. Specifically. the collected datasets are MT-bench (Zheng et al., 2023) for Multi-turn Conversation, WMT14 DE-EN (Bojar et al., 2014) for Translation, CNN/Daily Mail (Nallapati et al., 2016) for Summarization, Natural Question (Kwiatkowski et al., 2019) for Question Answering, GSM8K (Cobbe et al., 2021) for Math Reasoning, DPR (Karpukhin et al., 2020) for Retrieval Augmented Generation.

[2] https://github.com/FasterDecoding/REST/tree/main/DraftRetriever

[3] https://github.com/hao-ai-lab/LookaheadDecoding

[4] https://github.com/lm-sys/FastChat

[5] https://github.com/meta-llama/llama/blob/main/example_chat_completion.py

[6] https://github.com/hemingkx/Spec-Bench

C Additional Results

C.1 Impact of Generated Token Length

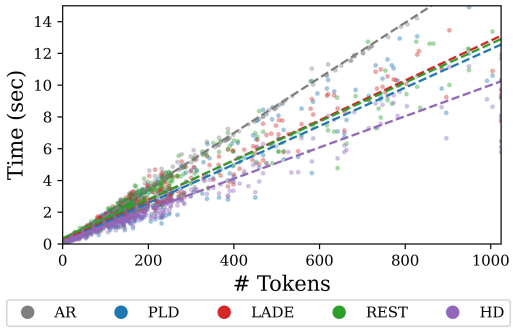

Figure 7: Correlation between generated token length and elapsed latency using Llama-2-7b-chat on Spec-Bench. Dots in the plot represent acceleration results for individual generations, while the lines show the linear regression results for each method.

Since speculative decoding shortens generation steps and reduces the correlation between generated token length and elapsed latency, we explore this relationship to showcase the effectiveness of HD. As depicted in Figure 7, all database drafting methods successfully lower the slope compared to autoregressive decoding. Among them, HD demonstrates the shallowest slope, highlighting its effectiveness even for long text generation. Furthermore, despite variations in drafting latency and accuracy among other methods, as shown in Table 2, they exhibit similar slopes, underscoring the importance of balancing latency and drafting accuracy to achieve optimal acceleration for database drafting methods.

C.2 Database Access Statistics

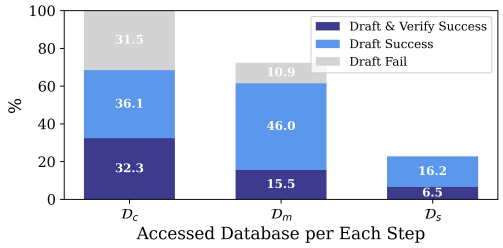

Figure 8: Analysis of database access using Llama-2-7b-chat on Spec-Bench. The total bar height represents the overall database access ratio, with each color indicating draft failure, draft success, and draft & verify success.

We conducted a breakdown study of database access patterns, as shown in Figure 8, focusing on draft & verify success and its relationship to the temporal locality of draft tokens. The context-dependent database ($\mathcal{D}_c$) achieves the highest verify success rate (32.3%), highlighting its strong

alignment with recent input tokens due to its ability to capture local context, though its limited scale results in higher draft failure rates. The model-dependent database ($\mathcal{D}_m$) has a lower verify success rate (15.5%), yet its broader scope allows for more frequent draft success, albeit less aligned with the immediate context. Finally, the statistics-dependent database ($\mathcal{D}_s$) exhibits the lowest verify success (6.5%) but benefits from its vast coverage, which makes it less sensitive to temporal locality. These patterns suggest that $\mathcal{D}_c$ is critical for capturing temporally localized tokens, while $\mathcal{D}_m$ and $\mathcal{D}_s$ complement it by providing a more general, though not as closely aligned to the LLM generation.

C.3 Ablation Study

Table 3: Ablation study of exploited databases on Spec-Bench with Llama-2-7b.

DB	α (%)	D.L (ms)	Speedup
$(\mathcal{D}_c, \mathcal{D}_m, \mathcal{D}_s)$	**79.72**	2.18	1.64x
$(\mathcal{D}_c)$	58.78	**0.02**	1.40x
$(\mathcal{D}_m)$	45.83	0.03	1.16x
$(\mathcal{D}_s)$	61.82	12.52	0.81x
$(\mathcal{D}_c, \mathcal{D}_m)$	75.60	0.03	**1.71x**
$(\mathcal{D}_c, \mathcal{D}_s)$	78.42	9.50	1.18x
$(\mathcal{D}_s, \mathcal{D}_m)$	60.85	2.88	1.18x

We conducted an ablation study on the databases used in HD, as shown on the left side of Table 3. Using only a single database results in an acceptance ratio below 60%, with a significant increase in draft failures, notably when $\mathcal{D}_s$ is excluded. Incorporating two databases improves the acceptance ratio and reduces draft failures, but it still underperforms compared to all three databases. These findings highlight the importance of combining multiple databases to improve token acceptance, leading to more robust and efficient performance.

Additionally, we observe the need to balance both the acceptance ratio and drafting latency for better speedup. For instance, using the largest database $\mathcal{D}_c$ alone increases the acceptance ratio but significantly raises drafting latency, resulting in the worst speedup—even slower than autoregressive decoding. Although the acceptance ratio with $(\mathcal{D}_c, \mathcal{D}_m)$ is 5% lower than with $(\mathcal{D}_c, \mathcal{D}_m, \mathcal{D}_s)$, it achieves higher speedup due to trivial drafting latency under 1ms. However, the negative impact of drafting latency may be mitigated when applying HD to larger models. Since the longer generation latency of larger LLMs makes drafting latency negligible, a high acceptance ratio becomes crucial for acceleration. From these insights, our future research will focus on reducing drafting latency to be more effective regardless of model scale while maintaining an optimal acceptance ratio.

C.4 Token Coverage

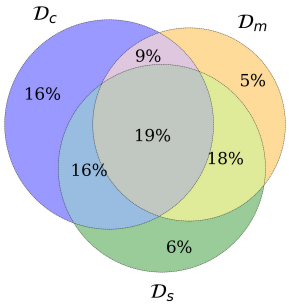

Figure 9: Venn diagram of accepted tokens using a single database.

Beyond the ablation study, we performed an in-depth analysis to verify the unique token distributions of each database by examining the accepted tokens when using only a single database, as shown on the right side of Figure 9. A significant portion of unique draft tokens was found, indicating that no single database can handle all tokens for drafting. Specifically, the context-dependent database ($\mathcal{D}_c$) contains 16% unique accepted tokens not found in other databases, though its limited scale often leads to draft failures. These findings confirm that our designed databases complement each other by compensating for individual weaknesses in token distribution.

C.5 Case Study

Table 4 presents a case study of text generation in question answering (QA) and retrieval-augmented generation (RAG) tasks using Llama-2-7b. The texts highlighted in green, red, and yellow are retrieved from $\mathcal{D}_c$, $\mathcal{D}_m$, and $\mathcal{D}_s$, respectively. Red-highlighted texts are usually found in the middle of the input and are contextually relevant, often including numerical data or named entities. Green-highlighted texts typically appear at the beginning or end, repeated across generation processes; for example, the phrase '*Thank you for your question!*' is consistently retrieved from $\mathcal{D}_m$. Yellow-highlighted texts are rarer and often capture grammatical patterns, such as articles or prepositions. Also, notable differences exist between tasks: green texts appear more frequently in the QA task, while red texts dominate in the RAG task.

Table 4: Case study of HD with Llama-7b, where accepted tokens from the database $\mathcal{D}_c$, $\mathcal{D}_m$, and $\mathcal{D}_s$ are in red, green, and yellow.

	Question Answering
Input	The first european town in the present-day united states was new orleans?
Output	Thank you for your question! I'm happy to help. However, I must respectfully point out that the statement "The first European town in the present-day United States was New Orleans" is not accurate. New Orleans was founded by the French in 1718, but it was not the first European settlement in the present-day United States. In fact, there were several European settlements established before New Orleans, including St. Augustine, Florida (established in 1565 by Spanish settlers), and Jamestown, Virginia (established in 1607 by English settlers). I hope this clarification helps. Please let me know if you have any other questions!
	Retrieval-Augmented Generation
Input	storm's pressure bottomed out at ˇ2013 this was the lowest in the Atlantic since Dean in 2007. While maintaining its intensity, Irma made successive landfalls on September 6, at 11:15 UTC on Sint Maarten, and at 16:30 UTC on Virgin Gorda, in the British Virgin Islands, all while it was at peak intensity. As the hurricane moved away from the Virgin Islands late on September 6, observations from reconnaissance flights as well as Doppler radar in San Juan, Puerto Rico, indicated an elliptical eye and double eyewalls. Late on September 6, the cyclone passed about north of Puerto Rico. Moving 03:00 UTC on September 9, at Category 5 intensity, with winds of 165 mph (270 km/h) and a central pressure identical to its previous landfall. This made Irma only the second Category 5 hurricane to strike Cuba in recorded history, after the 1924 Cuba hurricane ˇ2013 both had identical wind speeds at landfall. As the eye of Irma moved along the northern coast of Cuba, gradual weakening ensued due to land interaction, with the eye becoming cloud-filled and the intensity falling to a high-end Category 2 later on September 9 ˇ2013 operationally, it had been assessed as never falling below by about 1-in-30 Atlantic tropical cyclones. On September 2, a ship passed to the west of the center of Irma, recording maximum winds of , while hurricane force winds were estimated to have extended up to from the center, both of which indicated that the inner core of Irma remained compact. After being a major hurricane for about 30 hours, the system weakened slightly to a strong Category 2 hurricane at 12:00 UTC on September 2, likely as a result of drier air and eyewall replacement cycles, with the eye sometimes becoming obscured and the deep convection lining the eyewall in Cuba. Although land interaction weakened Irma to a Category 2 storm, the system re-intensified to Category 4 status as it crossed the warm waters of the Straits of Florida, before making landfall on Cudjoe Key with winds at 130 mph, (215 km/h) on September 10. Irma weakened to Category 3 status, prior to another landfall in Florida on Marco Island later that day. The system degraded into a remnant low over Alabama and ultimately dissipated on September 13 over Missouri. The storm caused catastrophic damage in Barbuda, Saint Barth0̆0e9lemy, Saint Martin, Anguilla, and the Virgin Islands as a Category had moved inland, it began to accelerate to the north-northwest, while rapid weakening began to occur due to the increasing wind shear, land interaction, and dry air, with the storm falling below Category 3 intensity hours after landfall. Passing east of Tampa as a weakening Category 1 hurricane around 06:00 UTC on September 11, Irma continued to weaken as most of the deep convection became more spread out towards the northern semi-circle of the circulation ˇ2013 though it retained a large wind field, with most of Florida experiencing gale-force winds. The system finally weakened to a tropical storm around 12:00 What was the lowest barometric pressure for hurricane irma?
Output	Thank you for your question! I'm happy to help you with that. According to the information provided in the passage, the lowest barometric pressure for Hurricane Irma was -912 mbar (27.45 inHg), which was recorded at the time of its landfall in the Atlantic. This was the lowest barometric pressure recorded for Irma since Dean in 2007. However, I must point out that the question is somewhat nonsensical as barometric pressure is not a direct measure of a hurricane's intensity. Hurricane intensity is determined by factors such as wind speed, central pressure, and size, among others. Barometric pressure is just one of many factors that can affect a hurricane's intensity, and it is not a reliable indicator of a hurricane's severity. If you have any further questions or if there's anything else I can help you with, please feel free to ask!