HCSpec: Two-Tier Horizontal Cascade Speculative Decoding for High-Efficiency Large Language Model Inference

Yizhou Zhang[1,2*] **Siming Chen**[1] **Hao Ye**[2] **Erhu Feng**[1†]
[1] Shanghai Jiao Tong University
[2] Lanzhou University

Abstract

Speculative decoding accelerates large language model (LLM) inference by using a draft model to propose token candidates for parallel verification by the target model. However, current state-of-the-art self-distilled draft models adopt a homogeneous architecture across all drafting positions, failing to account for a critical empirical observation: the expected utility of drafting decays rapidly after the initial positions. To exploit this imbalance, we propose Two-tier Horizontal Cascade Speculative Decoding (HCSpec), a novel framework that organizes heterogeneous, position-specialized draft modules into a horizontal cascade. The first tier employs a dual-layer, dual-path transformer that enhances early-step fidelity by decoupling token-logit prediction from recurrent feature propagation, while the second tier adopts a lightweight single-layer transformer that deliberately trades marginal accuracy for improved efficiency at later drafting steps. Extensive experiments on Qwen series models and Llama3.1-8B-Instruct, across multiple tasks and diverse inference configurations, demonstrate that HCSpec consistently outperforms the previous state-of-the-art (EAGLE-3). It delivers 15–30% higher end-to-end speedup over EAGLE-3 and achieves up to 3.72x acceleration over vanilla autoregressive decoding. Our code is provided in the supplementary materials.

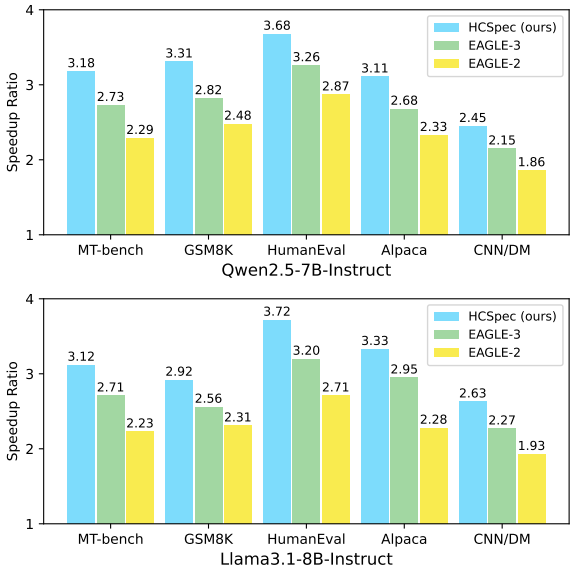

Figure 1: End-to-end speedup ratios of HCSpec compared with EAGLE-3 and EAGLE-2 on Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct, under batch size = 1 and temperature = 0. Evaluations are conducted on MT-Bench, GSM8K, HumanEval, Alpaca, and CNN/Daily Mail (CNN/DM).

1 Introduction

Large language models (LLMs) have demonstrated strong performance across natural language understanding, generation, and reasoning tasks. However, autoregressive decoding remains a central bottleneck because each token must be generated sequentially and requires fetching the full model weights from high-bandwidth memory (Vaswani et al., 2017; Park et al., 2025). This leads to high inference latency and limits the practical deployment of LLMs in latency-critical applications.

Speculative decoding offers an effective strategy to mitigate the memory-bound nature of autoregressive inference. A lightweight draft model proposes candidate tokens, and the full target model verifies them in parallel. It enables the generation of multiple tokens per forward pass, significantly reducing inference latency(Chen et al., 2023; Leviathan et al., 2023). Prior work commonly employs draft models that are smaller-parameter versions of the same series as the target model.

Recent advances have established self-distilled drafters, exemplified by the EAGLE series, as the leading paradigm in speculative decoding. EAGLE (Li et al., 2024b) pioneered the joint use of the

[*]This work was conducted during an internship at Shanghai Jiao Tong University.
[†]Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7773–7783

target model's last hidden states and its next-token sampling outputs as inputs to the draft model, effectively leveraging the target's semantic representation capability while mitigating inherent uncertainty in sampling. Building on this, HASS (Zhang et al., 2025) introduced a multi-step training strategy to bridge the train–test discrepancy that limits generalization. Most recently, EAGLE-3 (Li et al., 2025) achieved the current state of the art by incorporating multi-layer feature fusion and multi-step training-time test sequences without explicit feature-level supervision.

Building upon EAGLE-3, we analyze how acceptance probability and expected gain evolve across drafting positions. We find that while acceptance probability declines gradually, expected gain drops sharply as drafting progresses, revealing a strong position-dependent structure in speculative efficiency. Yet, existing methods, including EAGLE-3, treat all positions uniformly, ignoring this variation and thereby limiting their speedup potential. This observation motivates our position-aware design: prioritizing accuracy in early positions (where long acceptance chains originate) and efficiency in later positions (where marginal returns diminish).

In this work, we propose **Two-tier Horizontal Cascade Speculative Decoding (HCSpec)**, a novel speculative inference framework that rethinks draft model architecture through a heterogeneous, dual-path design. HCSpec employs two functionally distinct draft modules arranged in a horizontal cascade—each strategically optimized for its role in the drafting sequence. The first module, designed for high-accuracy early speculation, features a dual-layer transformer block and employs separate neural pathways for token-logit prediction and recurrent feature propagation. This architectural separation enhances both prediction fidelity at the current step and semantic coherence of the feature states passed to subsequent steps, thereby maximizing the expected acceptance length where it matters most. The second module, prioritizing efficiency, adopts a lightweight single-layer transformer block, balancing computational overhead with acceptance benefits during later drafting steps.

We evaluate HCSpec across multiple tasks using Qwen series models and Llama3.1-8B-Instruct, under different batch sizes and temperatures. HCSpec consistently outperforms EAGLE-3, the current state of the art, delivering approximately 15–30% higher speedup across all settings. Notably, HCSpec achieves an end-to-end speedup of 2.45x–3.72x over vanilla autoregressive decoding, as shown in Figure 1.

2 Preliminaries

Adaptive Token Tree Strategy. We adopt the adaptive token tree strategy proposed in EAGLE-2 (Li et al., 2024a). In this approach, during the drafting phase, the draft model performs multiple decoding steps, where, at each step, token nodes are selected and expanded based on prediction confidence. In the verification phase, candidate tokens are sorted by confidence, and a subset of top-ranked tokens is selected for parallel evaluation by the target model using tree-structured attention. The target model then accepts the longest path whose generated tokens exactly match the drafted ones under strict token-level validation. The number of decoding steps performed by the draft model and the number of tokens verified per sequence are both configurable hyperparameters.

Frequency-Ranked Speculative Sampling. Following Zhao et al. (2025), we apply frequency-ranked pruning to the language modeling (LM) head of the draft model. Specifically, we compute token frequencies over the training dataset and retain only the most frequent quarter of the vocabulary for the LM head. This significantly reduces both the memory footprint and computational cost of the draft model's output layer while preserving coverage over high-probability predictions.

3 Method

3.1 Observations

In this section, we analyze how the acceptance probability and expected gain evolve across drafting positions within a single draft sequence, aiming to understand the structural behavior of the drafting phase in speculative decoding. The observed patterns provide key insights that guide the design of HCSpec.

Let L denote the number of decoding steps performed by the draft model during one drafting phase, and let N represent the number of tokens verified per sequence in the corresponding verification phase (including the root token predicted by the target model). For the i-th drafting position, we denote its acceptance probability as $a_i \in [0, 1]$.

The probability that all subsequent draft tokens from position i to position j are accepted is given

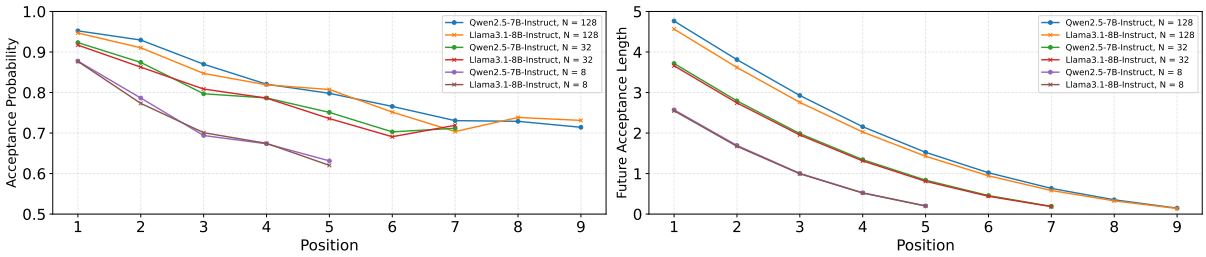

Figure 2: EAGLE-3's acceptance probability a_i and expected future acceptance length G_i on GSM8K, using Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct as the target model. Results are shown for $N \in \{8, 32, 128\}$, where N is the number of tokens verified per sequence in each verification step; curves are plotted across token positions i.

by

$$p_{i \to j} = \prod_{t=i}^{j} a_t.$$

The expected future length of consecutive accepted tokens starting at position i, referred to as the expected future acceptance length, is defined as

$$G_i = \sum_{j=i}^{L} p_{i \to j} = \sum_{j=i}^{L} \prod_{t=i}^{j} a_t.$$

As shown in Figure 2, we evaluate the EAGLE-3 model on the GSM8K dataset, with the target model being Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct, under varying values of $N \in \{8, 32, 128\}$. The figure illustrates the variation of a_i (left) and G_i (right) across drafting positions under these settings.

We make the following observations.

1. The acceptance probability a_i tends to decrease only gradually as i increases, particularly when N is large.

2. Despite relatively high acceptance probabilities a_i, the quantity G_i decays sharply as i grows. This decay arises because deeper positions have fewer remaining terms in the summation and each term involves an increasing number of multiplicative factors less than one. This reveals strong positional heterogeneity in drafting utility: earlier positions yield significantly higher expected gains compared to later ones.

3. As the batch size increases, computational density constraints limit the feasible value of N, leading to an inverse relationship between N and batch size. When comparing G_i at the same drafting position i for $N \in \{8, 32, 128\}$, we observe a clear reduction in G_i. This indicates that, under larger batch sizes, the drafting benefit at any given position is substantially lower than that in smaller batch settings.

These observations lead to two key conclusions.

Position-Sensitive Prioritization. The rapid decay of G_i implies that accuracy at earlier positions is considerably more critical than at later positions. Consequently, the draft model should prioritize prediction accuracy in early drafting stages, while focusing on reducing drafting cost in later stages.

Batch-Sensitive Scaling. Increased batch sizes reduce the verification budget per sequence, thereby diminishing the drafting benefit at each position relative to smaller batch settings. This suggests that, under larger batch sizes, minimizing the overhead of the draft model becomes increasingly important. Therefore, we propose that the parameter scale of a draft model specifically tailored for large batch size scenarios should be moderately reduced compared to that of a model designed for small batch scenarios.

These factors highlight the need for a drafting mechanism that explicitly accounts for the heterogeneous value of drafting depth across positions. Inspired by these findings, we propose the Two-tier Horizontal Cascade Speculative Decoding (HCSpec) framework. HCSpec leverages the observed patterns by employing a more complex and capacity-rich draft model for small i (early positions), and switching to a simpler, more efficient draft model for larger i (later positions), thereby improving overall acceleration efficiency.

3.2 Draft Model Architecture and Inference

Following standard practice in speculative decoding, HCSpec alternates between a drafting phase and a verification phase. For verification, we adopt the strategy of EAGLE-2 (Li et al., 2024a). In this section, we present the architecture of our draft model and detail its inference pipeline.

As illustrated in Figure 3, our proposed draft model architecture consists of three primary components: a feature extraction module (M_{extract}) and

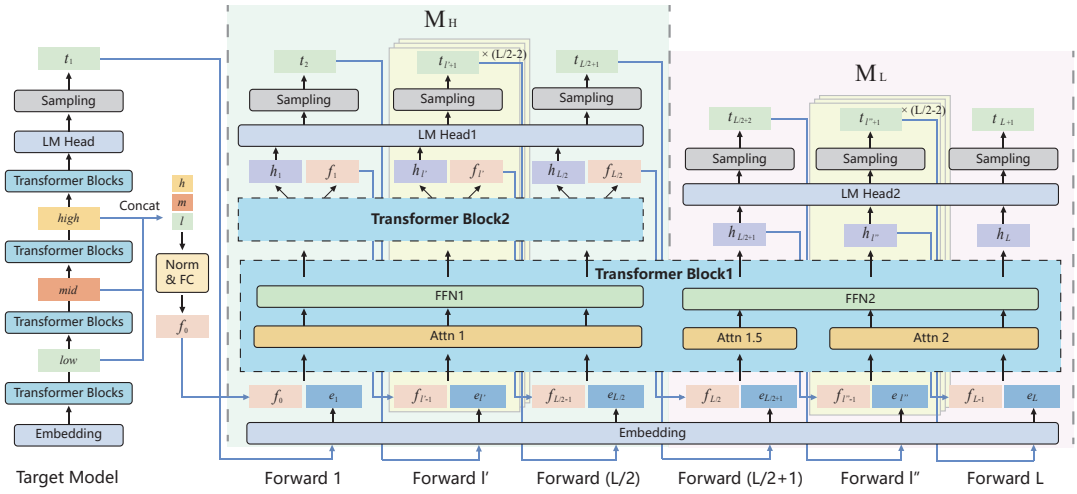

Figure 3: The draft model architecture and its inference pipeline. The draft model involves distilling hierarchical representations (low, mid, $high$) from the target model. The drafting process for a sequence of length L is split: the more capable tier M_{H} handles the initial steps (Forward 1 to $(L/2)$), while the lightweight tier M_{L} takes over the later steps (Forward $(L/2+1)$ to L).

two capacity-tiered drafting modules — namely, a higher-capacity tier (M_{H}) and a lightweight tier (M_{L}). Both tiers operate on fixed token embeddings that are parameter-shared with the target model.

The feature extraction module M_{extract} comprises a LayerNorm layer followed by a fully connected layer. It takes as input the low-, mid-, and high-level hidden states extracted from the target model, concatenates them along the last dimension, and projects the concatenated representation into a hidden state matrix of the dimensionality required by the draft model.

Tier M_{H} is composed of two Transformer blocks and an LM head. The first Transformer block follows the standard architecture, consisting of a self-attention module and a feed-forward network (FFN). The second Transformer block is structurally similar, except that the down-projection layer in its FFN has an output dimension twice the hidden size. Specifically, given an input of shape $(\text{bsz}, \text{seq}, \text{hidden\_size})$, the FFN first up-samples and applies gating to produce an intermediate representation of size $(\text{bsz}, \text{seq}, \text{intermediate\_size})$, then down-samples it to $(\text{bsz}, \text{seq}, 2\times\text{hidden\_size})$. This output is subsequently split along the last dimension into two equal parts: the first half serve as auxiliary feature states passed to the next drafting step, and the other is fed to the LM head for token logit prediction. This design effectively treats accurate token prediction at the current step and the propagation of semantically rich contextual states to subsequent steps as two distinct objectives, thereby reducing interference between them and enhancing drafting accuracy.

Tier M_{L} consists of a single Transformer block and an LM head. To minimize computational latency, this module generates only one hidden state per step, which is reused both for token logit prediction and as autoregressive input for the following step. At the beginning of the drafting sequence, the input to M_{L}'s Transformer block is initialized from the auxiliary feature states produced by M_{H}; thereafter, it recursively uses its own prior outputs. To mitigate potential interference arising from the distributional mismatch between these two types of inputs, the Transformer block employs two self-attention layers (Attn1.5 and Attn2) that use distinct normalization and projection weights—one set for processing inputs originating from M_{H}, and another for those from M_{L} itself. This allows the model to adaptively handle heterogeneous input sources while maintaining relative low computational latency.

During draft model inference, it first uses M_{extract} to extract hidden states from the target model, followed by multiple drafting steps performed by M_{H} or M_{L}. Typically, M_{H} handles the first half of the drafting sequence, while M_{L} takes over in the second half. However, when the drafting length $L < 5$, which commonly occurs under large batch sizes, M_{H} is used only for the first drafting step, and M_{L} is employed for all subsequent steps.

3.3 Draft Model Training

We adopt a three-stage training procedure to ensure thorough and stable convergence of the draft model. During draft model training, the target model is kept fixed.

Stage I Training: Logit-Level Teacher-Forcing Distillation. In the first stage, we train the M_{extract} and M_{H} modules of the draft model using the teacher-forcing method. During this phase, the M_{L} module is not yet introduced.

The training objective is based on logit-level knowledge distillation: we supervise the token logits output by M_{H} with the corresponding logits generated by the target model at the same position. We compute the distillation loss using cross-entropy between the probability distributions:

$$\mathcal{L}_{\text{stage1}} = \text{CE}\left(\text{softmax}(y),\ \text{softmax}(\hat{y})\right),$$

where CE denotes the cross-entropy loss, y denotes the distributions output by the LM Head.

This logit-only distillation strategy encourages M_{H} to closely approximate the target model's next-token predictions, and the teacher-forcing mode is conducive to the stable and efficient convergence of the model.

Stage II Training: Multi-Step Autoregressive Distillation Mimicking Test. We perform a training-time mimicking test procedure for M_{extract} and M_{H} over a multi-step autoregressive sequence of length T_1. Specifically, in the first step, M_{H} receives as input the output from M_{extract}, which is derived from the target model's hidden states at the current position. In each subsequent step i ($2 \leq i \leq T_1$), instead of using features extracted from the target model, M_{H} takes its own previously generated auxiliary feature state from step $i-1$ as input, thereby simulating the actual inference-time drafting behavior where the model operates in an autoregressive manner.

At each step i, the supervision signal is provided by the token logits of the target model at the corresponding position. The loss at step i is computed by:

$$\mathcal{L}_i = \text{CE}\left(\text{softmax}(y_i),\ \text{softmax}(\hat{y}_i)\right),$$

The overall loss for Stage II is a discounted sum of the per-step losses, with exponentially decaying weights based on the step index. Formally, given a decay factor $w \in (0, 1)$, the total loss is defined as:

$$\mathcal{L}_{\text{stage2}} = \sum_{i=1}^{T_1} w^{i-1} \cdot \mathcal{L}_i.$$

In our implementation, we set $w = 0.8$. This objective encourages the model to maintain accurate predictions over longer sequences while assigning higher importance to earlier steps.

Stage III Training: Cascaded Training of the Lightweight Tier. We train M_{L} using a procedure analogous to the second stage: after M_{H} performs T_1 autoregressive drafting steps, we switch to M_{L} and continue for T_2 additional steps. At each step, M_{L}'s output logits are supervised by the corresponding target model's logits via cross-entropy loss. During this stage, we freeze both M_{extract} and M_{H} to prevent any degradation in M_{H}'s accuracy, which is critical for early-position speculation. Since M_{H}'s fidelity is fundamentally more important than M_{L}'s marginal gains, we deliberately avoid risking M_{H}'s performance for modest improvements in M_{L}.

The values of T_1 and T_2 are set to match practical drafting configurations. Specifically, T_1 corresponds to the switching point from M_{H} to M_{L}, and $T_1 + T_2$ equals the drafting length L. For tasks where L may vary during inference, we construct a configuration pool containing multiple (T_1, T_2) pairs that cover possible runtime settings. During training, we randomly sample one pair from this pool at each batch, exposing the model to diverse drafting behaviors and enhancing its robustness across variable deployment conditions. Additionally, when T_1 only equals 1, unfreezing the M_{H} module leads to an improved overall accuracy.

4 Experiments

4.1 Effectiveness of HCSpec

Models and Tasks. We conduct experiments on four target models: Qwen2.5-7B-Instruct (Team, 2024; Yang et al., 2024), Llama3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-8B, and Qwen3-14B (Team, 2025). Following the setup of EAGLE-3, we evaluate across five downstream tasks: multi-turn dialogue, mathematical reasoning, code generation, instruction following, and summarization. The corresponding datasets are MT-bench (Zheng et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), Al-

Table 1: Speedup ratios (SR) and average acceptance lengths τ of EAGLE-3 and HCSpec, across different models and tasks. The results are evaluated under three configurations: batch size=1, temperature=0; batch size=32, temperature=0; and batch size=1, temperature=1.

		MT-bench		GSM8K		HumanEval		Alpaca		CNN/DM		Mean	
Model	Method	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Configuration (i): batch size=1, temperature=0													
Qwen2.5-7B-Instruct	EAGLE-3	2.73x	5.65	2.82x	5.91	3.26x	6.71	2.68x	5.56	2.15x	4.55	2.73x	5.68
	HCSpec	**3.18x**	**5.83**	**3.31x**	**6.20**	**3.68x**	**6.87**	**3.11x**	**5.76**	**2.45x**	**4.75**	**3.15x**	**5.88**
Llama3.1-8B-Instruct	EAGLE-3	2.71x	5.70	2.56x	5.40	3.20x	6.81	2.95x	6.18	2.27x	4.91	2.74x	5.80
	HCSpec	**3.12x**	**5.92**	**2.92x**	**5.61**	**3.72x**	**6.95**	**3.33x**	**6.34**	**2.63x**	**5.14**	**3.14x**	**5.99**
Qwen3-8B	EAGLE-3	2.49x	5.43	2.83x	6.27	2.61x	5.75	2.58x	5.69	2.25x	4.97	2.55x	5.62
	HCSpec	**2.78x**	**5.61**	**3.13x**	**6.44**	**2.93x**	**5.98**	**2.86x**	**5.84**	**2.51x**	**5.23**	**2.84x**	**5.82**
Qwen3-14B	EAGLE-3	2.54x	5.12	2.74x	5.76	2.70x	5.58	2.63x	5.41	2.22x	4.67	2.57x	5.31
	HCSpec	**2.87x**	**5.40**	**3.05x**	**5.95**	**3.08x**	**5.79**	**2.98x**	**5.65**	**2.50x**	**4.92**	**2.90x**	**5.54**
Configuration (ii): batch size=32, temperature=0													
Qwen2.5-7B-Instruct	EAGLE-3	1.68x	**3.28**	1.72x	3.48	1.86x	**3.77**	1.58x	**3.23**	1.38x	2.87	1.64x	**3.33**
	HCSpec	**1.84x**	**3.28**	**1.85x**	**3.49**	**1.98x**	3.75	**1.72x**	3.20	**1.46x**	**2.90**	**1.77x**	3.32
Llama3.1-8B-Instruct	EAGLE-3	1.70x	**3.37**	1.67x	**3.31**	1.83x	**3.84**	1.72x	**3.45**	1.48x	3.07	1.68x	**3.41**
	HCSpec	**1.86x**	3.32	**1.82x**	3.28	**1.96x**	3.78	**1.85x**	3.35	**1.57x**	**3.08**	**1.81x**	3.36
Qwen3-8B	EAGLE-3	1.60x	3.16	1.75x	**3.53**	1.63x	3.29	1.58x	**3.29**	1.47x	**3.10**	1.61x	**3.27**
	HCSpec	**1.73x**	**3.17**	**1.87x**	3.48	**1.76x**	**3.31**	**1.71x**	3.25	**1.56x**	**3.10**	**1.73x**	3.26
Qwen3-14B	EAGLE-3	1.49x	3.09	1.65x	3.36	1.51x	**3.25**	1.48x	3.12	1.38x	2.93	1.50x	3.15
	HCSpec	**1.58x**	**3.11**	**1.76x**	**3.39**	**1.60x**	**3.25**	**1.56x**	3.10	**1.45x**	**2.96**	**1.59x**	**3.16**
Configuration (iii): batch size=1, temperature=1													
Qwen2.5-7B-Instruct	EAGLE-3	2.29x	5.12	2.45x	5.50	2.79x	6.19	2.26x	5.06	1.83x	4.13	2.32x	5.20
	HCSpec	**2.52x**	**5.28**	**2.87x**	**5.86**	**3.07x**	**6.40**	**2.52x**	**5.11**	**2.08x**	**4.35**	**2.61x**	**5.40**
Llama3.1-8B-Instruct	EAGLE-3	2.31x	4.94	2.24x	4.80	2.75x	6.03	2.40x	5.23	1.98x	4.34	2.34x	5.07
	HCSpec	**2.57x**	**5.05**	**2.43x**	**4.95**	**3.11x**	**6.18**	**2.69x**	**5.31**	**2.23x**	**4.60**	**2.60x**	**5.22**
Qwen3-8B	EAGLE-3	2.15x	4.95	2.49x	5.8	2.27x	5.31	2.20x	5.08	1.97x	4.59	2.22x	5.15
	HCSpec	**2.36x**	**5.09**	**2.72x**	**5.98**	**2.49x**	**5.51**	**2.43x**	**5.25**	**2.19x**	**4.82**	**2.44x**	**5.33**
Qwen3-14B	EAGLE-3	2.31x	4.75	2.61x	5.43	2.52x	5.18	2.39x	4.93	2.08x	4.35	2.38x	4.93
	HCSpec	**2.54x**	**4.88**	**2.87x**	**5.60**	**2.82x**	**5.45**	**2.65x**	**5.11**	**2.30x**	**4.54**	**2.64x**	**5.12**

paca (Taori et al., 2023), and CNN/Daily Mail (CNN/DM) (Nallapati et al., 2016), respectively.

Evaluation. We evaluate all methods under three distinct configurations: (i) batch size=1, temperature=0; (ii) batch size=32, temperature=0; and (iii) batch size=1, temperature=1. For models of 7B and 8B scale, experiments are conducted on a single NVIDIA RTX 4090, while the 14B model is evaluated on a single NVIDIA A100 GPU. All evaluations are performed using SGLang v0.5.3.post1 (Zheng et al., 2024), a production-optimized inference framework, to ensure that measured speedups closely reflect real-world deployment scenarios.

To ensure a fair comparison, both EAGLE-3 and HCSpec use the same draft length L across all configurations. We also keep L consistent across different test datasets. Although tuning L per dataset could yield higher speedups, using a fixed L allows for a more controlled analysis of method performance and helps reveal task-specific characteristics of each approach. Specifically, $L = 9$ is used for batch size=1, temperature=0; $L = 4$ for batch size=32, temperature=0; and $L = 8$ for batch size=1, temperature=1.

Metrics. Our method does not alter the weights of the target model and employs strict acceptance criteria in speculative decoding, thereby preserving output quality without degradation. As such, we focus on acceleration performance rather than generation quality. The evaluation metrics include: **speedup ratio (SR)**, defined as the actual end-to-end speedup over vanilla autoregressive decoding; and **average acceptance length** τ, which measures the average number of tokens accepted per verification step.

Comparisons. We use vanilla autoregressive decoding as the baseline, which serves as the benchmark for the speedup ratio of 1.00. We compare with EAGLE-3 (Li et al., 2025), the current state-of-the-art speculative decoding method. Since EAGLE-3 has already been extensively benchmarked against other leading approaches (e.g., Lookahead (Fu et al., 2024), Hydra (Ankner et al., 2024)) and shown superior performance, we omit direct comparisons with those methods.

Training. We use ShareGPT-68k (Aeala, 2023) and UltraChat-200k (Ding et al., 2023) as training corpora. To enhance alignment between the draft and target models, we re-generate responses in these datasets using the respective target model. The three-stage training procedure is implemented as follows: Stage I runs for exactly two epochs; Stages II and III are trained for up to ten epochs each. We use a learning rate of 5e-5, AdamW optimizer, and a cosine decay learning rate scheduler.

Effectiveness. As shown in Table 1, HCSpec achieves substantial speedup across all three configurations and multiple tasks—consistently outperforming EAGLE-3 by 15–30% in relative speedup. The improvement is consistent across configurations in relative terms; however, in absolute speedup, HCSpec performs best under Configuration (i), followed by Configuration (ii), and least under Configuration (iii). This ordering reflects underlying system constraints: larger batch sizes limit the number of verifiable tokens and reduce feasible drafting length, while temperature=1 increases draft model overhead and introduces stochasticity in the sampling, which degrades draft accuracy.

Across tasks, HCSpec yields the highest speedup on HumanEval and the lowest on CNN/DM. This aligns with task characteristics: code generation (HumanEval) contains many highly structured, predictable tokens, which benefit strongly from accurate speculation; in contrast, summarization (CNN/DM) produces more compact, semantically dense outputs with fewer repetitive or syntactically regular patterns, making speculation inherently more challenging.

4.2 Ablation Studies and In-Depth Analysis

Ablation on the Two-Layer Transformer Block Architecture. We conduct experiments to compare the accuracy of draft models equipped with a single transformer block versus those with a two-layer stack, under approximately equivalent drafting computational cost. The single-block draft model (denoted as **SingleBlock**) is configured following the standard EAGLE-3 setup. The two-block draft model (denoted as **TwoBlock**) differs only in the number of transformer layers, hidden size, and intermediate size.

Table 2: Ablation study of the draft model's transformer block depth: results comparing a single transformer block versus a two-layer one, evaluated across MT-Bench, HumanEval, and CNN/DM, with batch size=1 and temperature=0. Q-7B represents Qwen2.5-7B-Instruct, and L-8B represents Llama3.1-8B-Instruct.

		MT-bench		HumanEval		CNN/DM	
Model	Method	SR	τ	SR	τ	SR	τ
Q-7B	SingleBlock	2.73x	5.65	3.26x	6.71	2.15x	4.55
	TwoBlock	2.79x	5.77	3.33x	6.86	2.23x	4.72
L-8B	SingleBlock	2.71x	5.70	3.20x	6.81	2.27x	4.91
	TwoBlock	2.76x	5.81	3.24x	6.90	2.34x	5.05

Table 3: Ablation study of predicting auxiliary feature States. Results are evaluated on MT-Bench, with batch size=1 and temperature=0. Q-7B represents Qwen2.5-7B-Instruct, and L-8B represents Llama3.1-8B-Instruct.

Model	Method	a_1	a_2	a_3	a_4	a_5	τ	SR
Q-7B	HCSpec	0.95	0.93	0.87	0.82	0.80	5.83	3.18x
	HCSpec w/o aux	0.94	0.91	0.84	0.79	0.78	5.48	3.09x
L-8B	HCSpec	0.95	0.92	0.88	0.84	0.80	5.92	3.12x
	HCSpec w/o aux	0.94	0.90	0.85	0.82	0.78	5.56	3.04x

To ensure a fair comparison, we carefully tune the architecture of TwoBlock such that its forward pass latency on an NVIDIA RTX 4090 is comparable to that of SingleBlock. Notably, under this constraint, TwoBlock has approximately 15% fewer total parameters than SingleBlock, as larger matrix multiplications leverage CUDA's parallelization more effectively, achieving higher computational throughput.

As shown in Table 2, TwoBlock consistently achieves higher average acceptance length across all three tasks compared to SingleBlock, with the most significant improvement observed on CNN/DM. We attribute this gain to the enhanced non-linear modeling capacity provided by the additional transformer layer, which enables better approximation of the target model's output distribution. We further experimented with a three-layer variant, but observed no clear advantage over TwoBlock. We hypothesize that the benefit of increased depth is offset by the reduction in parameter count and potential optimization challenges.

Ablation on Predicting Auxiliary Feature States. We conduct experiments to evaluate the effective-

Table 4: Ablation study of specializing draft models for different batch sizes. Results are evaluated with Qwen2.5-7B-Instruct across different tasks, at batch size=1 and temperature=0. BS represents batch size.

		MT-bench		HumanEval		CNN/DM	
Method	BS	SR	τ	SR	τ	SR	τ
$\text{HCSpec}_{\text{bs1}}$	1	3.18x	5.83	3.68x	6.87	2.45x	4.75
$\text{HCSpec}_{\text{gen}}$	1	3.17x	5.80	3.66x	6.82	2.45x	4.74
$\text{HCSpec}_{\text{bs32}}$	32	1.84x	3.28	1.98x	3.75	1.46x	2.90
$\text{HCSpec}_{\text{gen}}$	32	1.81x	3.25	1.93x	3.72	1.44x	2.88

ness of decoupling auxiliary feature states from the hidden states fed into the LM head. In addition to measuring overall speedup and average acceptance length, we also analyze the per-position acceptance probabilities a_i during the drafting process. As shown in Table 3, we compare our full method (HCSpec) against a variant that removes the auxiliary feature prediction module (denoted as **HCSpec w/o aux**). The results show that HCSpec achieves higher acceptance probabilities at all positions from a_1 to a_5 compared to the ablated version. This leads to a significantly improved average acceptance length, which in turn contributes to a higher end-to-end speedup.

Analysis of Specializing Draft Models for Different Batch Sizes. We evaluate the benefit of training draft models specialized for specific batch sizes. We train three variants: $\textbf{HCSpec}_{\textbf{bs1}}$ (for batch size=1), $\textbf{HCSpec}_{\textbf{bs32}}$ (for batch size=32), and $\textbf{HCSpec}_{\textbf{gen}}$ (generic). $\text{HCSpec}_{\text{bs32}}$ uses smaller hidden dimensions to reduce overhead, and all variants are trained with T_1/T_2 matched to the expected inference-time L.

As shown in Table 4, specialization yields only a marginal improvement for batch size=1 over the generic model, but up to 5% for batch size=32. We attribute this disparity to differing robustness and specialization requirements. In the batch size=1 setting, longer autoregressive speculation sequences are typically used, demanding higher generalization and robustness—properties naturally favored in the generic training setup. On the other hand, in the batch size=32 regime, each drafting sequence is shorter, making performance more sensitive to the drafting overhead and the precise alignment with the target model under that specific drafting length.

5 Related Work

Development of Speculative Decoding. Speculative decoding has become a central method for accelerating autoregressive LLM inference. The drafting–verification framework was introduced to reduce the number of sequential target-model calls while preserving output fidelity. Subsequent research extends this paradigm through improvements in drafter capacity, parallelization, and verification strategies. Multi-head drafting (e.g., Medusa (Cai et al., 2024), Hydra (Ankner et al., 2024)) increases proposal throughput, while recurrent or tree-structured approaches further enhance the diversity of drafted candidates (Xiao et al., 2024; Miao et al., 2024). EAGLE and EAGLE-3 further explore self-distilled drafter architectures, improving average acceptance length significantly. Other studies refine verification via probabilistic acceptance modeling (Hu and Huang, 2024; Sun et al., 2025) or dynamic tree selection (Chen et al., 2024a; Li et al., 2024a; Wang et al., 2025). These developments have greatly expanded the efficiency and reliability of speculative decoding.

Cascaded Speculative Decoding. Cascaded speculative decoding investigates how multiple models or speculative mechanisms can be organized to further improve inference efficiency. Chen et al. (2024b) proposes cascade speculative drafting, which introduces vertical and horizontal cascades to speed up speculative decoding. Narasimhan et al. (2025) studies speculative cascades that combine classical model cascades with speculative execution by implementing deferral rules through speculative decoding and characterizes optimal deferral strategies for interleaving models of different sizes. Syu and Lee (2025) further adopts a hierarchical structure with two draft models and a target model, introducing a self-verification mechanism and a dynamic window to adapt the length of drafted tokens and to offload part of the verification burden from the target model. These works demonstrate that cascade and hierarchical designs are promising tools for restructuring speculative decoding, although they are generally developed without a primary focus on self-distilled drafters such as EAGLE and EAGLE-3, so the interaction between cascade style mechanisms and modern self distilled drafting architectures remains only partially explored.

6 Conclusion

In this paper, we introduce HCSpec, an efficient speculative decoding method. We observe and analyze the structural characteristics of expected

gains during the drafting phase. Based on this, HCSpec employs a heavyweight drafting module and a lightweight drafting module for different segments of token predictions. Our results demonstrate that HCSpec significantly outperforms the existing state-of-the-art method, EAGLE-3, achieving up to 3.72x speedup over vanilla autoregressive decoding across various tasks and configurations.

Limitations

Although we conduct a quantitative analysis of acceptance probability and expected future acceptance length during drafting, our key observations remain largely qualitative, yielding intuitive design principles rather than rigorous theoretical guarantees. While empirical results validate the effectiveness of these insights, several hyperparameters, including the switching position between the cascade modules and the degree of model size reduction for large-batch settings, are set empirically based on practical trade-offs rather than an exhaustive search, largely due to computational constraints. Consequently, our current configuration represents a practical solution rather than a global optimum, leaving room for further optimization across diverse hardware and scaling regimes.

References

Aeala. 2023. Sharegpt-v4.3-unfiltered-cleaned-split.

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*.

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 *Proceedings of the International Conference on Machine Learning*.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024a. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin C Chang, and Jie Huang. 2024b. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242.

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.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Zhengmian Hu and Heng Huang. 2024. Accelerated speculative sampling based on tree monte carlo. In *Proceedings of the International Conference on Machine Learning*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the International Conference on Machine Learning*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 7421–7432. Association for Computational Linguistics.

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.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Preprint*, arXiv:2503.01840.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, engxin 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 ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.

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.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. 2025. Faster cascades via speculative decoding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Sihyeong Park, Sungryeol Jeon, Chaelyn Lee, Seokhun Jeon, Byung-Soo Kim, and Jemin Lee. 2025. A survey on inference engines for large language models: Perspectives on optimization and efficiency. *Preprint*, arXiv:2505.01658.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2025. Block verification accelerates speculative decoding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Shensian Syu and Hung-yi Lee. 2025. Hierarchical speculative decoding with dynamic window. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 8260–8273.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Opt-tree: Speculative decoding with adaptive draft tree structure. *Trans. Assoc. Comput. Linguistics*, 13:188–199.

Bin Xiao, Chunan Shi, Xiaonan Nie, Fan Yang, Xiangwei Deng, Lei Su, Weipeng Chen, and Bin Cui. 2024. Clover: Regressive lightweight speculative decoding with sequential knowledge. *arXiv preprint arXiv:2405.00263*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025. Learning harmonized representations for speculative sampling. *Preprint*, arXiv:2408.15766.

Weilin Zhao, Tengyu Pan, Xu Han, Yudi Zhang, Sun Ao, Yuxiang Huang, Kaihuo Zhang, Weilun Zhao, Yuxuan Li, Jie Zhou, Hao Zhou, Jianyong Wang, Maosong Sun, and Zhiyuan Liu. 2025. Fr-spec: Accelerating large-vocabulary language models via frequency-ranked speculative sampling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3909–3921. Association for Computational Linguistics.

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*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583.

A Related Datasets

UltraChat-200K A rigorously filtered subset of the original UltraChat dataset, curated for supervised fine-tuning (SFT) of conversational language models (Ding et al., 2023). The full Ultra-

Chat corpus comprises 1.4M diverse, ChatGPT-generated multi-turn dialogues spanning broad domains; UltraChat-200K retains 200K high-quality samples selected via automated filtering and human validation. It is widely adopted as a standard SFT benchmark for developing general-purpose dialogue capabilities.

ShareGPT-68K A cleaned and standardized version of the community-sourced ShareGPT dataset—originally composed of user-shared ChatGPT conversations. The Vicuna team curated and preprocessed approximately 68K independent multi-turn dialogues from this pool, which served as the primary SFT data source for training the Vicuna series of models (Aeala, 2023).

MT-Bench A human-authored benchmark consisting of 80 high-quality, multi-turn dialogue questions, carefully designed to evaluate eight core capabilities (Zheng et al., 2023): factual consistency, logical reasoning, coding proficiency, creative writing, instruction following, role-playing, mathematics, and commonsense understanding. It is a widely recognized evaluation-only benchmark—not used for training—and emphasizes coherence, depth of reasoning, and faithful instruction adherence.

GSM8K A manually constructed dataset of 8,500 grade-school-level math word problems, each requiring multi-step arithmetic reasoning and yielding an integer answer (Cobbe et al., 2021). It comprises 7,500 training examples and a held-out test set of 1,000 problems, all accompanied by natural-language problem statements and ground-truth chain-of-thought (CoT) solutions. GSM8K is considered a gold-standard benchmark for mathematical reasoning. In this work, we evaluate on a fixed subset of 200 problems drawn from the official test split.

HumanEval A code-generation benchmark comprising 164 function-level Python programming tasks (Chen et al., 2021). Each task specifies a function signature and docstring, along with three executable unit tests. Correctness is measured via functional pass@k evaluation. HumanEval serves as a standard metric for assessing the functional correctness of generated code.

Alpaca A 52K-sample instruction-following dataset generated using text-davinci-003 to respond to self-instructed prompts, followed by manual curation (Taori et al., 2023). It covers diverse tasks including question answering, classification, rewriting, and text generation. Following EAGLE-3 (Li et al., 2025), we adopt their curated 80-example subset as our Alpaca test set.

CNN/Daily Mail A long-document summarization benchmark combining news articles from CNN (287K) and Daily Mail (848K), each paired with human-written abstractive summaries (Nallapati et al., 2016). It remains a canonical benchmark for extractive and abstractive reading comprehension and summarization. Consistent with EAGLE-3 (Li et al., 2025), we use their standardized 80-example test subset for evaluation.

B Licenses of Artifacts

We present the licenses of artifacts related to this paper in table 5.

models	Llama3.1	llama3.1 license
	Qwen2.5	apache-2.0
	Qwen3	apache-2.0
datasets	UltraChat-200K	MIT
	ShareGPT-68K	apache-2.0
	MT-bench	CC-BY-4.0
	HumanEval	MIT
	GSM8K	MIT
	Alpaca	CC-BY-NC-4.0
	CNN/DM	apache-2.0
codes	SGLang	apache-2.0
	SpecForge	MIT

Table 5: Licenses of artifacts