Multi-Drafter Speculative Decoding with Alignment Feedback

Anonymous ACL submission

Abstract

Speculative decoding (SD) accelerates large language model (LLM) inference by using a smaller model to draft future tokens, which are then verified by the target LLM. This preserves generation quality by accepting only aligned tokens. However, individual drafters, often trained for specific tasks or domains, exhibit limited effectiveness across diverse applications. To address this, we introduce METASD, a unified framework that integrates multiple drafters into the SD process. METASD dynamically allocates computational resources to heterogeneous drafters by leveraging alignment feedback and framing drafter selection as a multi-armed bandit problem. Extensive experiments show METASD consistently outperforms single-drafter approaches.

1 Introduction

Large language models (LLMs) like GPT-4 (Achiam et al., 2023), Gemini (Google et al., 2023), and Llama (Touvron et al., 2023) have significantly advanced applications such as search (Reid et al., 2024), coding assistance, and virtual assistants. However, their token-by-token generation process often results in substantial inference times, primarily due to memory bandwidth limitations (Patterson, 2004; Shazeer, 2019). Speculative decoding (SD) has emerged as a key technique to mitigate this, employing a smaller 'drafter' model to predict future tokens for parallel verification by the target LLM (Leviathan et al., 2023; Chen et al., 2023). This reduces latency by accepting only tokens aligned with the target model's predictions, thereby preserving output quality.

Existing SD methods have improved draft acceptance rates via architectural and training innovations (Liu et al., 2023; Zhou et al., 2023; Cai et al., 2024; Miao et al., 2024; Sun et al., 2023), including multi-path exploration through batched inference or tree verification (Sun et al., 2023; Miao et al., 2024; Cai et al., 2024) and better drafter-target alignment via knowledge distillation (Zhou et al., 2023; Liu et al., 2023). However, these methods predominantly rely on a single drafter, a design choice that curtails versatility. This limitation has become particularly significant with the growing trend towards deploying ensembles of specialized models, a paradigm actively explored in LLM routing (Hu et al., 2024; Jitkrittum et al., 2025). Consequently, SD with a single drafter can fail on out-of-distribution inputs or dynamic user queries where its inherent biases become a bottleneck (Liu et al., 2023; Yi et al., 2024).

Table 1: Comparison of METASD with existing single- and multi-drafter SD approaches. METASD is training-free without extra compute for drafting and verification cost than standard, offers theoretical regret guarantees and Robustness to Non-Stationarity (RNS).

Method	Multi-drafter	Training-free	No extra compute	Regret bound	RNS
Standard SD	✗	✓	—	—	—
Static Ensemble SD	✓	✓	✗	✗	✓
Learned Routing	✓	✗	✓	✗	✗
METASD (Ours)	✓	✓	✓	✓	✓

The limitations of single-drafter systems necessitate an adaptive, multi-drafter framework capable of generalizing across diverse and evolving user queries. This is compounded by operational constraints, including the need for low computational overhead, scalability across variable user loads, and robustness to traffic fluctuations without manual intervention or pre-tuning. To meet these requirements, SD systems must incorporate dynamic adaptation mechanisms that can efficiently allocate computational resources while optimizing performance for diverse input conditions.

This paper addresses these challenges by introducing METASD, a novel framework that integrates multiple drafters into SD and dynamically meta-drafts the optimal drafter during inference. At the core of our framework is the concept of alignment feedback, which measures the compatibility

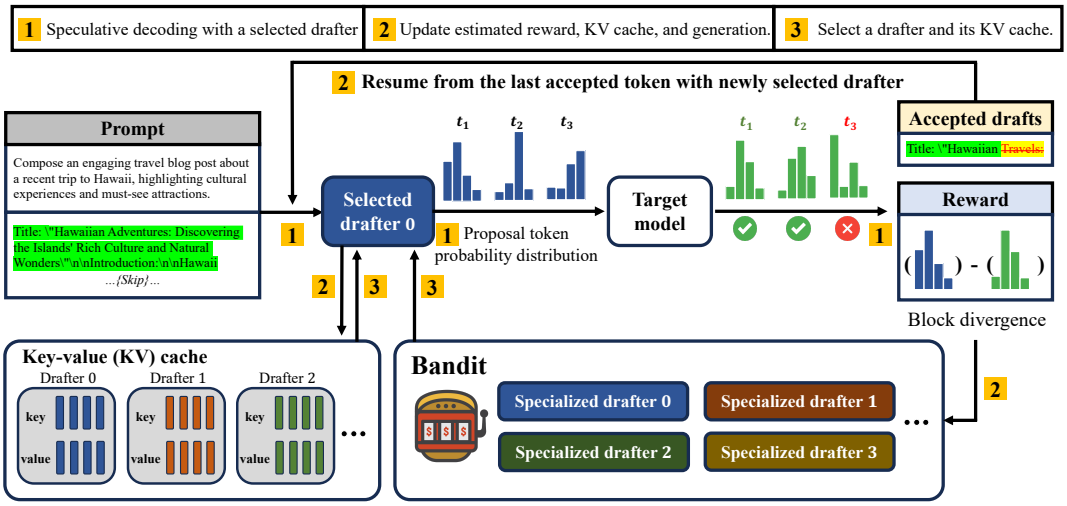


Figure 1: Overview of speculative decoding with multiple drafters in multi-armed bandit (MAB) framework. Drafters are selected and updated based on alignment feedback through the block divergence reward. The example in this figure is from an instance in MT-Bench dataset (Zheng et al., 2024).

between a drafter's predictions and the target LLM. Leveraging this feedback, METASD formulates drafter selection as an exploration-exploitation tradeoff (Gittins et al., 2011), dynamically learning a policy that selects the best drafter based on the current observation at every state. Inspired by multi-armed bandit (MAB) algorithms widely used in recommendation systems to adaptively optimize decisions based on feedback (Silva et al., 2022), our approach employs a similar mechanism to dynamically moderate between drafters. Unlike static configurations, METASD enables the system to adapt to both evolving input contexts and varying task distributions, achieving robust performance across scenarios (Figure 1). Our contributions are:

- **METASD**: A unified framework integrating multiple drafters into SD. MetaSD utilizes block divergence (BD) as a reward signal, derived from alignment feedback between drafter predictions and the target LLM, to enable dynamic drafter selection. It supports both black-box (independent drafters) and white-box (drafters using target LLM latent representations) configurations (**Section 2**).
- **Theoretical analysis**: We establish regret upper bounds for METASD, offering insights into its convergence, scalability, and the role of BD-based alignment feedback in optimizing drafter selection (**Section 3**).
- **Performance improvement**: Extensive experiments demonstrate METASD achieves superior inference speedups over existing single-drafter and static multi-drafter methods (**Section 4**).

2 Problem statement

2.1 Motivation

Speculative decoding (SD) employs a *draft-verify-accept* paradigm for faster inference. A drafter $\mathcal{M}_q$, which is smaller than the target LLM $\mathcal{M}_p$, drafts the future tokens $\{x^{l+1:l+N_{max}}\}$ based on the input sequence $x^{1:l}$. The target LLM assesses each token x^{l+j} $(j = 1, \ldots, N_{max})$ to determine whether $p(\cdot|x^{1:l+j-1})$ is aligned with its own predictions $q(\cdot|x^{1:l+j-1})$. Only the tokens aligned with the LLM's own predictions are accepted, ensuring the lossless generation (detailed in Appendix D).

Despite advancements, existing SD often employs a single drafter, leading to critical limitations in dynamic, multi-task settings. Task-specific drafters, though effective in-domain, generalize poorly to unseen tasks due to inherent training data biases (Yi et al., 2024; Liu et al., 2023). SD's efficacy hinges on the alignment between the drafter $\mathcal{M}_q$ and the target LLM $\mathcal{M}_p$; strong alignment yields higher acceptance rates. This alignment provides crucial real-time feedback for system adaptation. Consequently, addressing dynamic and multi-task challenges requires a framework that uses alignment feedback to adapt its drafter selection policy based on the current input context.

Motivated by these limitations, our METASD framework continuously evaluates alignment feedback to identify the optimal drafter for the current context. We frame this as a decision-making problem under uncertainty, leveraging the MAB framework. MAB offers a principled approach to balance exploration (evaluating different drafters) and ex-

Algorithm 1: MetaSD

INPUT : Drafters $[K]$, initial prompt $x^{1:l}$, target model $\mathcal{M}_p$, target sequence length B.

$t \leftarrow 0$
while $l < B$ **do**
  Meta-draft the drafter i in drafter pool $[K]$ following the bandit
  Execute one SD step with drafter i and target model given $x^{1:l}$
  Compute the BD between drafter i's prediction and $\mathcal{M}_p$'s prediction as the reward (Section 2.3)
  Update the sequence length with the number of accepted tokens from the draft $N_{acc}(i,t)$: $l, t \leftarrow l + N_{acc}(i,t) + 1, t + 1$
  Update the bandit
end while

ploitation (using the historically best-performing drafter) (Algorithm 1).

2.2 Problem formulation

Multi-armed bandit MAB framework addresses an online learning scenario where, at each round t, an agent takes an action by choosing an arm $a_t \in [K]$ and receives a reward r_t from the environment. The goal of MAB is to design an algorithm that maximizes the expectation of cumulative reward $\mathbb{E}[\sum_{t=1}^{T} r_t]$ throughout a total of T rounds. To achieve this, one can aim to design an optimal policy $\pi^\star$ to minimize the pseudo-regret, defined as: $\textsc{Reg}(\pi, T) = \sum_{t=1}^{T} \mathbb{E}[r_{a_t^\star}] - \mathbb{E}[r_{a_t}]$. Here, a_t denotes the action chosen in round t by the policy π and $a_t^\star$ represents the optimal action in round t which yields the highest expected reward. For a more comprehensive review, we refer the reader to Lattimore and Szepesvári (2020).

METASD: Multi-drafter SD We formalize multi-drafter integration in SD as a MAB problem, where each SD cycle (drafting, verifying, accepting tokens) constitutes one MAB round (Algorithm 1). At each round t, a drafter a a_t is chosen from a pool of K heterogeneous drafters, and an SD step is performed, yielding $N_{acc}(a_t, t)$ accepted tokens. The process ends once B tokens are generated. Unlike standard MAB settings with a fixed number of total rounds T, METASD's total rounds is stochastic, contingent on the chosen drafter's alignment efficiency and token acceptance rates. Interestingly, conventional regret minimization does not directly optimize SD efficiency (formal proof in Section G.3). To address this, we introduce a novel objective in the following.

Definition 1 (Stopping time regret)**.** *Denote $\tau(\pi, B)$ as the number of total rounds of bandit policy π with target sequence length B and $\pi^\star$ as the optimal policy which satisfies $\pi^\star = \arg\min_\pi \mathbb{E}[\tau(\pi, B)]$. Then, regret objective of MetaSD with policy π becomes:*

$$\textsc{Reg}(\pi, B) = \mathbb{E}\left[\tau(\pi, B)\right] - \mathbb{E}[\tau(\pi^\star, B)]\,. \quad (1)$$

The following lemma establishes this new regret objective as a valid performance metric for SD.

Lemma 1. *Minimizing $\textsc{Reg}(\pi, B)$ is equivalent to maximizing expected number of accepted tokens, which aligns with the objective of SD.*

Proof. B is sum of the total rounds and accepted tokens, $B = \tau(\pi, B) + \sum_{t=1}^{\tau(\pi,B)} N_{acc}(a_t, t)$. □

Consequently, minimizing the regret in eq. 1 directly maximizes the expected number of accepted tokens, aligning with the SD objective. Thus, the goal of MetaSD is to find a policy π that minimizes the regret in Definition 1.

Note that our algorithm operates in a lossless framework; **therefore, our evaluation focuses exclusively on inference speedups and acceptance rates.**

2.3 Alignment feedback as a reward

To dynamically select the best-aligned drafter in METASD without prior knowledge, the MAB algorithm requires informative reward signals reflecting the alignment quality between a given drafter and the target model. Intuitively, a well-aligned drafter will produce next-token probability distributions closely matching those of the target model for the same context. The following assumption formalizes this concept.

Assumption 1. *Denoting d_{TV} as total variation distance(TV distance), define $\alpha_{i,t}$ to be:*

$$\alpha_{i,t} = 1 - d_{TV}\left(p^t(\cdot|x^{1:t-1}), q_i^t(\cdot|x^{1:t-1})\right), \quad (2)$$

Then, for any instance of $x^{1:B}$ generated by the target model, $\alpha_{i,t}$'s are drawn i.i.d. from a distribution ν_i with expectation α_i. In other words, following holds for all $i \in [K]$.

$$\mathbb{E}[\alpha_{i,t}] = \alpha_i, \quad \alpha_{i,t} \overset{i.i.d.}{\sim} \nu_i. \quad (3)$$

Although seemingly strong, above assumption generalizes prior work in the speculative decoding literature, which typically assumes a fixed TV distance at every decoding step (Leviathan et al., 2023; Li et al., 2024). Furthermore, we empirically demonstrate in Section G.6 that the alignment between the drafter and verifier shows stationary behavior in most practical scenarios.

Definition 2 (Block divergence reward)**.** *Let t be the current round, i be the drafter index, and $l(t)$ be the number of input tokens for the target model at round t. Given context $x^{1:l(t)}$, denote $d_{TV}(p^{l(t)}, q_i^{l(t)}) = \frac{1}{2}\|p^{l(t)} - q_i^{l(t)}\|_1$ as the total variation (TV) distance between probability measures of target model ($p^{l(t)}$) and the drafter ($q_i^{l(t)}$). Then, we define the BD reward as:*

$$r_{i,t}^{BD} = \frac{1}{N_{max}} \sum_{j=0}^{N_{max}-1} \left(1 - d_{TV}\left(p^{l(t)+j}, q_i^{l(t)+j}\right)\right). \tag{4}$$

Another straightforward option for feedback signal is the block efficiency (BE), which quantifies the number of mean accepted tokens until a given round (Sun et al., 2023; Chen et al., 2023; Kim et al., 2024). Formally, for drafter i in round t, it is defined as: $r_{i,t}^{BE} := N_{acc}(i,t)/N_{max}$, where N_{max} is predefined maximum draft length and $N_{acc}(i,t)$ is number of accepted tokens.

Both BE and BD rewards act as alignment feedback in MetaSD, as shown by the following lemma:

Lemma 2. *BE and BD rewards are related as:*

$$\mathbb{E}[r_{i,t}^{BE}] = \frac{1 - \alpha_i^{N_{max}}}{N_{max}(1 - \alpha_i)} \mathbb{E}[r_{i,t}^{BD}]. \tag{5}$$

Consequently, maximizing BD reward is equivalent to maximizing number of accepted tokens.

The proof is in Section G.2. While both rewards are viable, we show that BD reward is superior in SD, both theoretically and empirically. The following theorem formalizes this comparison:

Theorem 1 (Informal)**.** *Under the stationary environment, for any reward design r_i with $\mu_i = \mathbb{E}[r_i]$, $i^\star = \arg\max \alpha_i$, and $\Delta_i = \mu_i^\star - \mu_i$, we define the feedback signal for each suboptimal arm $i \neq i^\star$ as*

$$R(r_i) := \frac{\Delta_i^2}{\max(\mathrm{Var}[r_i], \mathrm{Var}[r_{i^\star}])}. \tag{6}$$

Then, $R(r_i^{BD}) > R(r_i^{BE})$ for most cases.

Table 2: Comparison of reward statistics on the Japanese queries with multilingual specialized drafters. Detailed descriptions are in Table 9.

Drafter	Mean reward		variance		Zero-reward ratio†
	BE	BD	BE	BD	BE only
Ja-drafter	0.232	0.488	0.093	0.044	0.503
Ru-drafter	0.099	0.294	0.032	0.026	0.678
De-drafter	0.081	0.317	0.024	0.032	0.721
Fr-drafter	0.074	0.288	0.023	0.029	0.743
Zh-drafter	0.106	0.326	0.037	0.034	0.681

† Zero-reward ratio applies only to BE, as BD rewards are always nonzero.

Theorem 1 indicates that the BD reward offers a more informative feedback signal than BE. The feedback signal $R(r_i)$ (eq. 6) is crucial for bandit algorithm performance: a stronger signal enables faster and more accurate identification of the optimal arm, thereby minimizing exploration of suboptimal arms. Consequently, employing the BD reward can lead to superior performance in bandit-based drafter selection. A formal statement of Theorem 1 with a statistical analysis of both rewards (Lemma 5, Lemma 6) is in Section G.2.

We empirically validate Theorem 1 by analyzing the reward statistics gathered from the full speculative execution. In Table 2, the BD reward exhibits significantly larger differences in mean rewards between the optimal and suboptimal drafters (larger Δ_i), and consistently lower variance than BE. This provides a stronger feedback signal R, enabling more stable learning and faster convergence (details in Section F.6).

Concurrent works Hou et al. (2025) have leveraged the idea of MAB for SD, but different scope and contributions from ours: first, while Hou et al. (2025) modify the confidence range of the UCB algorithm to control the regret analysis estimating mean of number of accepted tokens, our theoretical analysis **can be applied to more general reward shaping scenarios**, resulting in stronger instance-dependent bound in Theorem 2. Furthermore, we reveal that reward design is important and choice of BD reward improves the performance of SD both practically and theoretically (Section E.3).

3 Method

This section presents MetaSD-UCB designed to optimize bandit-based drafter selection in MetaSD by leveraging alignment feedback via the BD reward.

3.1 Algorithm

MetaSD-UCB MetaSD-UCB (algorithm 2) adapts the Upper Confidence Bound (UCB)

Algorithm 2: MetaSD-UCB

INPUT : Drafter pool $[K]$, initial prompt $x^{1:l}$, target sequence length B, exploration parameter β.

1: $t \leftarrow 0$
2: **for** $i \in [K]$ **do**
3: Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 4)
4: $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t + 1$
5: **end for**
6: **while** $l < B$ **do**
7: $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta \sqrt{\frac{2 \ln t}{n_i}}$
8: Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 4)
9: $\hat{\mu}_{a_t,t} \leftarrow \frac{\hat{\mu}_{a_t,t} * n_{a_t} + r_{a_t,t}}{n_{a_t}+1}$
$n_{a_t}, l, t \leftarrow n_{a_t} + 1, l + N_{acc}(a_t,t) + 1, t + 1$
10: **end while**

algorithm (Auer, 2002) for SD, leveraging the BD reward to enhance convergence speed. Following the UCB principle, MetaSD-UCB selects drafters based on their empirical mean reward augmented by an uncertainty term (Line 7 in Algorithm 2), thus inherently balancing the exploration-exploitation trade-off.

3.2 Regret upper bound for MetaSD-UCB

While the UCB algorithm achieves optimal performance in standard stochastic bandit settings (Lattimore and Szepesvári, 2020), extending this guarantee to MetaSD is non-trivial. MetaSD's unique structure, particularly its novel stopping time regret objective (Definition 1), requires a dedicated analysis. We demonstrate that MetaSD-UCB attains a logarithmic regret bound with respect to the target sequence length B, formalized as follows.

Theorem 2 (Regret upper bound)**.** *Under Assumption 1 and MetaSD-UCB with the BD reward, there exists a constant $C, C' > 0$ such that following bound holds:*

$$\text{REG}(\pi, B) < \sum_{i \neq i^\star} \frac{8}{(N_{max})\Delta(\alpha_i)^2} (\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta(\alpha_i)^2})) + C') + C. \tag{7}$$

Here, we denote $\Delta(\alpha_i) = \alpha_{i^\star} - \alpha_i$, where $i^\star$ is the index of the optimal drafter. In Section G.5, we show the above regret bound is tighter then the regret bound of using BE reward. The improvement in eq. 7 stems directly from adopting the BD reward in algorithm 2. Since the number of observations within each round grows with N_{max}, the variance of the BD reward is effectively reduced by a factor of N_{max}. This, in turn, leads to a smaller constant term in the regret upper bound compared to using the BE reward. The following corollary captures this observation:

Corollary 1 (Informal)**.** *In most scenarios, the regret upper bound in eq. 7 is tighter than the regret upper bound obtained when using the BE reward with MetaSD-UCB.*

A complete proof of Theorem 2 and a formal statement of Corollary 1 with the proof are in Section G.5. Note that above analysis incorporates both temperature sampling and greedy decoding scenarios in Section G.6. Note that Theorem 2 is token generation instance dependent bound, holding for any specific realization of the token generation process. This per-realization guarantee is stronger than a bound on the expected regret over the distribution of all possible generations (Section G.7).

3.3 Extensions of MetaSD framework

Switching costs Switching between drafters incurs a computational cost due to KV-cache recalculations. Unlike classical bandits with fixed per-switch costs, the overhead in MetaSD depends on the number of unprocessed tokens in the current block. For this case, we propose the variant using Sequential Halving (SH) (details in Section H.1).

Non-stationary environment Non-stationary environments, where the optimal drafter may change across queries or even within a single generation , present further challenges for SD. MetaSD inherently handles inter-query non-stationarity through its query-by-query re-initialization. For intra-query non-stationarity, where the best drafter shifts mid-generation, MetaSD can be extended with non-stationary bandit algorithms (Kocsis and Szepesvári, 2006). Such algorithms track changing reward distributions, allowing MetaSD to sustain performance in dynamic settings. Section H.2 offers further discussion and experiments.

4 Experiment

4.1 Experimental setup

Models We use Vicuna 7B (Chiang et al., 2023) as the target LLM for both black-box and white-box SD. The primary difference between these approaches lies in the drafter's access to the target LLM: black-box drafters operate independently using only the target's final logits, while white-box drafters can access its intermediate activations and hidden states. For black-box SD, Vi-

Table 3: (Black-box SD) Speedup ratio relative to standard autoregressive greedy decoding on various datasets, comparing single specialized independent drafters, other methods (PLD (Saxena, 2023) and Lookahead (Fu et al., 2024)), and bandit-based drafter selection (Rand (uniformly random), EXP3 (Auer et al., 2002), SH (Karnin et al., 2013), UCB). Evaluations are conducted with a single NVIDIA A6000 GPU under greedy decoding settings. Drafter specializations: 1: Code, 2: Translation, 3: Summarization, 4: QA, 5: Math.

Speedup	SpS with specialized drafters					SpS	Other methods		Bandit in MetaSpS			
	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	OFA	PLD	Lookahead	Rand	EXP3	SH	UCB
Code	**2.437** ●	1.224	1.565	1.814	1.687	**2.435** ●	1.923	1.542	1.640	1.919	2.148	2.300
Trans	0.991	**2.076** ●	1.000	1.019	0.950	1.032	1.076	1.133	1.150	1.217	1.422	**1.587** ●
Sum	1.513	1.087	**2.133** ●	1.510	1.387	1.526	**2.501** ●	1.275	1.429	1.606	1.812	1.971
QA	1.332	1.200	1.343	**1.960** ●	1.252	1.267	1.178	1.208	1.294	1.437	1.599	**1.711** ●
Math	1.483	1.228	1.378	1.486	**2.454** ●	1.571	1.653	1.533	1.471	1.690	2.144	**2.280** ●

Table 4: (White-box SD) Speedup ratio relative to standard autoregressive greedy decoding on various datasets, comparing single specialized drafters, other methods (blockwise parallel decoding (BPD) (Stern et al., 2018), Medusa, Rescored-BPD (R-BPD) and Rescored-Medusa (Kim et al., 2024)), and bandit-based drafter selection. Evaluations are conducted with a single NVIDIA A100 GPU under greedy decoding settings.

Speedup	Specialized Eagle drafters					Eagle	Other methods				Bandit in MetaEagle			
	Eagle1	Eagle2	Eagle3	Eagle4	Eagle5	OFA	BPD	R-BPD	Medusa	R-Medusa	Rand	EXP3	SH	UCB
Code	**3.934** ●	1.303	1.776	2.150	2.427	**3.776** ●	1.963	2.146	2.661	2.822	2.310	2.858	3.650	3.724
Trans	1.750	**2.496** ●	2.281	2.131	1.714	2.143	1.626	1.442	1.909	2.056	2.036	2.171	2.225	**2.318** ●
Sum	1.707	1.507	**3.382** ●	2.005	1.589	2.640	1.509	1.455	1.723	2.136	2.261	2.261	2.801	**3.057** ●
QA	1.842	1.579	2.181	**2.916** ●	1.783	2.446	1.489	1.468	1.817	2.154	2.006	2.128	2.466	**2.641** ●
Math	2.584	1.618	2.337	2.433	**3.903** ●	3.049	1.696	1.696	2.142	2.519	2.449	2.811	3.339	**3.520** ●

Table 5: Speedup ratio relative to standard AR greedy decoding on various multilingual datasets, comparing single specialized drafters, all drafting method, and our bandit-based algorithm (EXP3, UCB). Evaluations are conducted with a single NVIDIA A5000 GPU under greedy decoding settings. Drafter specializations: 1: Ja →En, 2: Ru →En, 3: De →En, 4: Fr →En, 5: Zh →En.

Speedup	Sps with specialized drafters					Bandit in MetaSps		
	Drafter1	**Drafter2**	**Drafter3**	**Drafter4**	**Drafter5**	**DraftAll**	**EXP3**	**UCB**
Ja →En	**1.757** ●	1.109	1.012	1.018	1.154	0.352	**1.260** ●	1.161
Ru →En	1.055	**1.817** ●	0.995	0.963	1.036	0.396	1.259	**1.503** ●
De →En	1.098	1.369	**2.360** ●	1.036	1.099	0.544	1.472	**1.693** ●
Fr →En	1.106	1.445	1.108	**2.135** ●	1.122	0.484	1.506	**1.775** ●
Zh →En	1.198	1.086	1.021	1.023	**1.516** ●	0.367	1.204	**1.369** ●

cuna 68M (Yang et al., 2024) serves as the base drafter architecture; each such drafter is trained on distinct task-specific datasets (generated via self-distillation from the target LLM, following (Kim and Rush, 2016; Zhou et al., 2023; Cai et al., 2024; Yi et al., 2024)) to ensure heterogeneity. For white-box SD, we integrate Eagle (Li et al., 2024) with Vicuna 7B. Multiple Eagle drafters, sharing the same base architecture, are fine-tuned on separate task-specific datasets, also produced through self-distillation. As a baseline, we include a One-Size-Fits-All (OFA) drafter trained on a mixed dataset spanning all tasks, enabling evaluation of whether a single generalist drafter can match task-specialized ones. Further details are in Appendix F.

Number of drafts N_{max} For black-box SD, we employ speculative sampling (SpS) (Chen et al., 2023), generating one draft candidate per drafter, termed as MetaSpS. For multi-draft methods like Medusa (Cai et al., 2024) and Eagle (Li et al., 2024), we adhere to their original settings with a tree-attention mechanism. We employ the same tree structure for multiple Eagle drafters described in (Li et al., 2024), termed as MetaEagle. Unless otherwise stated, all approaches utilize a maximum of 5 drafts ($N_{max} = 5$).

Evaluation We conduct experiments on NVIDIA A5000, A6000, and A100 GPUs. To ensure adaptability, particularly in multi-turn conversations, the bandit algorithm is re-initialized for every new query. Our evaluation encompasses two primary scenarios: (1) a diverse task suite comprising coding (Code) from MT-Bench (Zheng et al., 2024), summarization (Sum) on CNN/Daily (Hermann et al., 2015), German-English translation (Trans) on WMT16 (Bojar et al., 2016), natural question answering (QA) (Kwiatkowski et al., 2019), and mathematical reasoning (Math) on GSM8K (Cobbe et al., 2021); and (2) multilingual translation tasks as detailed in Table 9, following Yi et al. (2024). For all evaluations, test datasets are randomly shuffled to simulate a non-stationary environment.

4.2 Main result

Table 3 presents speedup ratios for black-box SD across diverse tasks. As expected, specialized drafters perform best on their respective tasks but

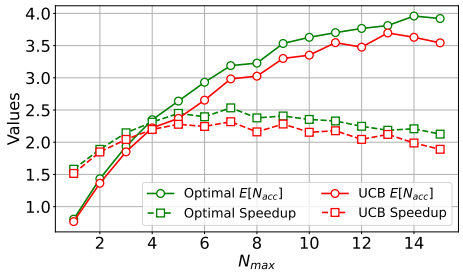


Figure 2: Ablations on N_{max}. 'Optimal' represents the optimal drafter and UCB denotes MetaSps-UCB with BD.

degrade significantly on unrelated ones, highlighting the limitations of static selection. Our MetaSpS-UCB consistently achieves competitive speedup across all tasks, often matching or surpassing both specialized drafters and state-of-the-art speculative decoding techniques. This demonstrates the effectiveness of adaptive drafter selection, allowing MetaSpS-UCB to dynamically leverage the most suitable drafter for a given input. While the OFA drafter performs well, MetaSD outperforms OFA in most cases, reinforcing the advantage of task-specialized selection over a single generalized model. Among bandit algorithms, MetaSpS-UCB demonstrates superior performance compared to other baselines (EXP3, SH), even when accounting for switching costs and non-stationarity. This empirically supports our theoretical analysis, confirming the advantages of UCB in SD.

Table 4 presents results for white-box SD using MetaEagle, where EAGLE drafters are integrated into the target LLM. Similar to MetaSpS case, specialized drafters perform well in their designated tasks but fail to generalize. In contrast, MetaEagle-UCB consistently achieves high speedup ratios, adapting effectively to different task distributions. This results demonstrates that our MetaSD algorithm can be applied to different SD architectures.

Speedup ratios on multilingual tasks, shown in Table 5, further corroborate these findings. As a baseline, we include DraftALL method, where in each round, all drafters are utilized for drafting. While this method generates optimal candidate at every round, verification increases linearly, resulting in significant decrease in speed-up ratios. In contrast, MetaSpS-UCB significantly outperforms this naive baseline, outperforming alternative bandit strategy (EXP3).

4.3 Comparing with MoE routing

Classification-based routing, analogous to Mixture-of-Experts (MoE) systems where experts (drafters) are selected via encoder outputs (e.g., BERT; Devlin et al. (2018)), serves as another baseline. Table 6 shows that such fine-tuned classifiers achieve competitive speedups on in-domain tasks. However, their performance substantially degrades on perturbed prompts and out-of-domain (OOD) tasks, where MetaSD maintains robustness. This disparity arises because MoE routing relies on input-level embeddings for a static drafter assignment, struggling with prompt variations and novel task distributions. Conversely, MetaSD dynamically adapts drafter selection at each decoding step, utilizing the Block Divergence (BD) reward to capture real-time, token-level alignment. These findings demonstrates that MetaSD's dynamic, alignment-driven selection consistently surpasses static classification-based approaches in complex scenarios.

Table 6: Speedup ratio comparison of MoE routing and MetaSD across different scenarios.

Routing Method	In-domain		Perturbed prompts		Out-of-Domain (Sps/Eagle)	
	SpS	Eagle	SpS	Eagle	RAG	Finance
MOE routing (BERT)						
Finetuned BERT-Base	1.772	2.759	1.456	2.245	1.645 / 2.132	1.410 / 2.413
Finetuned BERT-Large	1.741	2.797	1.567	2.323	1.638 / 2.110	1.398 /2.426
MetaSD (Bandit-based policy routing)						
MetaSpS-UCB	**1.912** ●	—	**1.820** ●	—	**1.799** ●	**1.436** ●
MetaEagle-UCB	—	**3.052** ●	—	**2.912** ●	**2.238** ●	**2.517** ●

Table 7: Average of speedup ratio comparing the BE and BD rewards for MetaSD-UCB with both SpS and EAGLE drafters over 3 different runs.

Task	MetaSpS-UCB		MetaEagle-UCB	
	BE	BD	BE	BD
Code	$2.052_{\pm 0.004}$	$2.231_{\pm 0.006}$ ●	$3.590_{\pm 0.017}$	$3.661_{\pm 0.003}$ ●
Trans	$1.465_{\pm 0.004}$	$1.554_{\pm 0.001}$ ●	$2.228_{\pm 0.009}$ ●	$2.201_{\pm 0.001}$
Sum	$1.770_{\pm 0.002}$	$1.929_{\pm 0.001}$ ●	$3.038_{\pm 0.005}$	$3.043_{\pm 0.001}$ ●
QA	$1.591_{\pm 0.003}$	$1.698_{\pm 0.001}$ ●	$2.629_{\pm 0.003}$ ●	$2.608_{\pm 0.001}$
Math	$1.992_{\pm 0.003}$	$2.238_{\pm 0.002}$ ●	$3.461_{\pm 0.009}$	$3.515_{\pm 0.001}$ ●

4.4 Ablation study

Draft length To analyze the impact of draft length on the performance of MetaSps-UCB with the BD reward, we conduct experiments on the Code task using 5 drafters following the same setting in Table 3, where we vary the draft length N_{max}. Figure 2 shows that increasing the draft length initially leads to higher $\mathbb{E}[N_{acc}]$ and speedup due to the increased parallelism in token generation. However, beyond a certain threshold, further increasing the draft length yields diminishing returns and can even decrease performance due to the higher probability of rejection and the associated overhead.

Reward design We compare MetaSD with different types of rewards. As shown in Table 7, BD

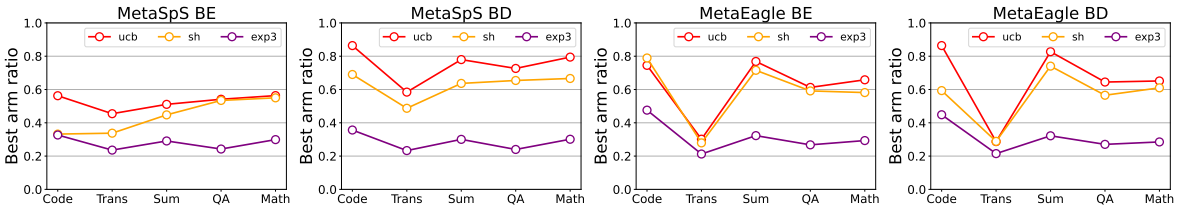


Figure 3: Best arm ratio over rounds for various configurations. (Left) MetaSpS (black-box SD) with BE and BD rewards. (Right) MetaEagle (white-box SD) with BE and BD rewards.

Table 8: White-box performance with perturbed prompts (speedup relative to greedy decoding on A100).

Task	Eagle1	Eagle2	Eagle3	Eagle4	Eagle5	OFA	MetaSD-UCB
Code	3.748	1.335	1.697	2.030	2.451	**3.626** ●	3.563
Trans	1.757	2.553	2.161	2.035	1.677	2.293	**2.375** ●
Sum	1.671	1.529	3.084	1.939	1.639	2.648	**2.742** ●
QA	1.837	1.616	2.094	2.932	1.722	2.439	**2.516** ●
Math	2.511	1.664	2.207	2.913	3.844	3.136	**3.366** ●

consistently outperforms BE in both MetaSps and MetaEagle scenarios, while the performance gap is less pronounced in MetaEagle-UCB. We attribute this difference to Eagle's tree-attention mechanism, which explores multiple decoding paths and implicitly mitigates the limitations of BE.

Robustness to rephrased prompts To simulate real-world conditions, we assess MetaSD's performance on perturbed prompts: queries slightly rephrased by GPT-4o while retaining their original meaning. These GPT-4o generated variations introduce natural linguistic shifts both across and within tasks. For instance, a translation prompt like 'Translate German to English' might become 'Convert this text from German to English' and 'Summarize:' could be rephrased as 'Provide a concise overview of the following text:'. While such prompt perturbations generally degrade the performance of all evaluated methods—underscoring the limitations of static drafter selection—MetaSD demonstrates robust performance, consistently outperforming baselines. This robustness stems from MetaSD's dynamic, token-level adaptation at inference, which mitigates the impact of prompt variations by not relying solely on initial prompt characteristics (details in Table 8 and Table 12).

Best arm ratio We evaluate the best arm ratio, which measures how frequently the optimal drafter is selected. Figure 3 tracks the evolution of this ratio across decoding rounds, comparing different reward functions and bandit algorithms for both MetaSpS and MetaEagle. Across all settings, UCB consistently converges to the optimal drafter faster than other bandit algorithms, with this effect being particularly pronounced in the MetaSpS setting. Additionally, the BD reward leads to a higher best arm ratio than BE. This aligns with our earlier findings that BD better captures the alignment dynamics in SD.

Further long-context experiments and stochastic decoding (temperature > 0) are in Appendix F.

5 Discussion

Memory bandwidth bound A potential concern with MetaSD is increased memory storage from loading multiple drafter models. However, our approach introduces minimal memory overhead. Storing all drafter weights in GPU DRAM mitigates frequent, slow system memory accesses—a key LLM bottleneck. For example, with a 7B target LLM and float16 precision, our MetaEagle framework uses at most 19GB of GPU DRAM, compared to 17GB for a single Eagle drafter. This minor increase in storage does not translate to higher memory bandwidth requirements during inference, as only one drafter is active and accessing VRAM at any time.

Throughput & efficiency MetaSD sustains high throughput efficiency, even in distributed, batch-processing environments. Table 17 shows MetaEagle-UCB surpasses the OFA Eagle baseline (e.g., 2.427 vs. 2.235 speedup). This performance is rooted in optimized drafter management: preloading parameters into DRAM minimizes memory transfers, ensuring consistent memory bandwidth irrespective of using single or multiple drafters. While MetaEagle-UCB remains competitive in heterogeneous batch settings (Table 17), a slight throughput reduction occurs due to increased I/O from drafter switching. Nevertheless, MetaSD prioritizes latency reduction for multi-task SD over raw throughput, as faster individual responses, even at moderately lower batch efficiency, typically enhance user experience more than higher-latency processing on a fully utilized GPU. MetaSD also features efficient KV cache management with minimal switching overhead. Eagle drafters, for instance, compute only a single KV cache layer for new tokens, making prefilling lightweight (Section F.11).

Limitations

While our framework supports general LLM architectures, broader evaluation is needed to evaluate its robustness across various models. The current single-query design can be extended to batched inference, enabling efficient instance-level drafter selection for mixed-task batches. Also, computation overhead might increase with a larger number of drafters. We provide a detailed discussion of these limitations and potential solutions in Appendix C.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Rajeev Agrawal. 1995. Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. Advances in applied probability, 27(4):1054–1078.

Idan Amir, Guy Azov, Tomer Koren, and Roi Livni. 2022. Better best of both worlds bounds for bandits with switching costs. Advances in neural information processing systems, 35:15800–15810.

Jean-Yves Audibert and Sébastien Bubeck. 2010. Best arm identification in multi-armed bandits. In COLT-23th Conference on learning theory-2010, pages 13–p.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. 2007. Tuning bandit algorithms in stochastic environments. In International conference on algorithmic learning theory, pages 150–165. Springer.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. 2009. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. Theoretical Computer Science, 410(19):1876–1902.

P Auer. 2002. Finite-time analysis of the multiarmed bandit problem.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multi-armed bandit problem. SIAM journal on computing, 32(1):48–77.

Peter Auer, Pratik Gajane, and Ronald Ortner. 2019. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In Conference on Learning Theory, pages 138–158. PMLR.

Ali Baheri and Cecilia O Alm. 2023. Llms-augmented contextual bandit. arXiv preprint arXiv:2311.02268.

Loïc Barrault, Ondřej Bojar, Marta R Costa-Jussa, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, and 1 others. 2019. Findings of the 2019 conference on machine translation (wmt19). In Findings of the 2019 conference on machine translation (WMT19). ACL.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. 2014. Stochastic multi-armed-bandit problem with non-stationary rewards. Advances in neural information processing systems, 27.

Gautam Bhartia. 2023. Finance alpaca dataset. Accessed: 2024-11-20.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, and 1 others. 2014. Findings of the 2014 workshop on statistical machine translation. In Proceedings of the ninth workshop on statistical machine translation, pages 12–58.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, and 1 others. 2016. Findings of the 2016 conference on machine translation (wmt16). In First conference on machine translation, pages 131–198. Association for Computational Linguistics.

Sébastien Bubeck. 2010. Bandits games and clustering foundations. Ph.D. thesis, Université des Sciences et Technologie de Lille-Lille I.

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. arXiv preprint arXiv:2401.10774.

Alexandra Carpentier and Andrea Locatelli. 2016. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In Conference on Learning Theory, pages 590–604. PMLR.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

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.

Lijie Chen, Jian Li, and Mingda Qiao. 2017. Towards instance optimal bounds for best arm identification. In Conference on Learning Theory, pages 535–592. PMLR.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion

Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168.

Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. 2014. Bandits with switching costs: T 2/3 regret. In Proceedings of the forty-sixth annual ACM symposium on Theory of computing, pages 459–467.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. 2013. Multi-armed bandit with budget constraint and variable costs. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 27, pages 232–238.

Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab Mirrokni. 2021. Regret bounds for batched bandits. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35(8), pages 7340–7348.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2002. Pac bounds for multi-armed bandit and markov decision processes. In Computational Learning Theory: 15th Annual Conference on Computational Learning Theory, COLT 2002 Sydney, Australia, July 8–10, 2002 Proceedings 15, pages 255–270. Springer.

Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. 2006. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. Journal of machine learning research, 7(6).

Nicolò Felicioni, Lucas Maystre, Sina Ghiassian, and Kamil Ciosek. 2024. On the importance of uncertainty in decision-making with large language models. arXiv preprint arXiv:2404.02649.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. arXiv preprint arXiv:2402.02057.

Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. 2019. Batched multi-armed bandits problem. Advances in Neural Information Processing Systems, 32.

Aurélien Garivier and Emilie Kaufmann. 2016. Optimal best arm identification with fixed confidence. In Conference on Learning Theory, pages 998–1027. PMLR.

Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In International conference on algorithmic learning theory, pages 174–188. Springer.

John Gittins. 1974. A dynamic allocation index for the sequential design of experiments. Progress in statistics, pages 241–266.

John Gittins, Kevin Glazebrook, and Richard Weber. 2011. Multi-armed bandit allocation indices. John Wiley & Sons.

Google, Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, and 1 others. 2023. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. Proceedings of the International Conference on Learning Representations (ICLR).

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR).

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. Advances in neural information processing systems, 28.

Marco Heyden, Vadim Arzamasov, Edouard Fouché, and Klemens Böhm. 2024. Budgeted multi-armed bandits with asymmetric confidence intervals. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 1073–1084.

Junya Honda and Akimichi Takemura. 2010. An asymptotically optimal bandit algorithm for bounded support models. In COLT, pages 67–79. Citeseer.

Yunlong Hou, Fengzhuo Zhang, Cunxiao Du, Xuan Zhang, Jiachun Pan, Tianyu Pang, Chao Du, Vincent YF Tan, and Zhuoran Yang. 2025. Banditspec: Adaptive speculative decoding via bandit algorithms. arXiv preprint arXiv:2505.15141.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. arXiv preprint arXiv:2403.12031.

Jerry Huang, Prasanna Parthasarathi, Mehdi Rezagholizadeh, and Sarath Chandar. 2024. Context-aware assistant selection for improved inference acceleration with large language models. arXiv preprint arXiv:2408.08470.

Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. 2014. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In Conference on Learning Theory, pages 423–439. PMLR.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, and 1 others. 2025. Universal model routing for efficient llm inference. arXiv preprint arXiv:2502.08773.

Zohar Karnin, Tomer Koren, and Oren Somekh. 2013. Almost optimal exploration in multi-armed bandits. In International conference on machine learning, pages 1238–1246. PMLR.

Taehyeon Kim, Ananda Theertha Suresh, Kishore Papineni, Michael Riley, Sanjiv Kumar, and Adrian Benton. 2024. Exploring and improving drafts in blockwise parallel decoding. Preprint, arXiv:2404.09221.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. arXiv preprint arXiv:1606.07947.

Levente Kocsis and Csaba Szepesvári. 2006. Discounted ucb. In 2nd PASCAL Challenges Workshop, volume 2, pages 51–134.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:453–466.

Tze Leung Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. Advances in applied mathematics, 6(1):4–22.

Tor Lattimore and Csaba Szepesvári. 2020. Bandit algorithms. Cambridge University Press.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, pages 19274–19286. PMLR.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large scale language model society. Preprint, arXiv:2303.17760.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th international conference on World wide web, pages 661–670.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. arXiv preprint arXiv:2401.15077.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient llm reasoning. arXiv preprint arXiv:2501.19324.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. arXiv preprint arXiv:2406.03853.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Ion Stoica, Zhijie Deng, Alvin Cheung, and Hao Zhang. 2023. Online speculative decoding. arXiv preprint arXiv:2310.07177.

Shie Mannor and John N Tsitsiklis. 2004. The sample complexity of exploration in the multi-armed bandit problem. Journal of Machine Learning Research, 5(Jun):623–648.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 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, pages 932–949.

Makoto Morishita, Katsuki Chousa, Jun Suzuki, and Masaaki Nagata. 2022. Jparacrawl v3. 0: A large-scale english-japanese parallel corpus. arXiv preprint arXiv:2202.12607.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. arXiv preprint arXiv:2406.18665.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. 2025. Adaptive llm routing under budget constraints. arXiv preprint arXiv:2508.21141.

Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. 2024. Do llm agents have regret? a case study in online learning and games. arXiv preprint arXiv:2403.16843.

David A Patterson. 2004. Latency lags bandwith. Communications of the ACM, 47(10):71–75.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv e-prints.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding

across millions of tokens of context. arXiv preprint arXiv:2403.05530.

Chloé Rouyer, Yevgeny Seldin, and Nicolo Cesa-Bianchi. 2021. An algorithm for stochastic and adversarial bandits with switching costs. In International Conference on Machine Learning, pages 9127–9135. PMLR.

Apoorv Saxena. 2023. Prompt lookup decoding.

ShareGPT. 2023. Sharegpt: Vicuna unfiltered dataset. https://huggingface.co/datasets/Aeala/ShareGPT_Vicuna_unfiltered. Accessed: 2024.

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

Chengshuai Shi, Kun Yang, Zihan Chen, Jundong Li, Jing Yang, and Cong Shen. 2024. Efficient prompt optimization through the lens of best arm identification. arXiv preprint arXiv:2402.09723.

Nícollas Silva, Heitor Werneck, Thiago Silva, Adriano CM Pereira, and Leonardo Rocha. 2022. Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions. Expert Systems with Applications, 197:116669.

Aleksandrs Slivkins and Eli Upfal. 2008. Adapting to a changing environment: the brownian restless bandits. In COLT, pages 343–354.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. Advances in Neural Information Processing Systems, 31.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. Spectr: Fast speculative decoding via optimal transport. arXiv preprint arXiv:2310.15141.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Long Tran-Thanh, Archie Chapman, Enrique Munoz De Cote, Alex Rogers, and Nicholas R Jennings. 2010. Epsilon–first policies for budget–limited multi-armed bandits. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 24, pages 1211–1216.

Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas Jennings. 2012. Knapsack based optimal policies for budget–limited multi–armed bandits. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 26, pages 1134–1140.

Peter Whittle. 1988. Restless bandits: Activity allocation in a changing world. Journal of applied probability, 25(A):287–298.

Fanzeng Xia, Hao Liu, Yisong Yue, and Tongxin Li. 2024a. Beyond numeric awards: In-context dueling bandits with llm agents. arXiv preprint arXiv:2407.01887.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024b. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. arXiv preprint arXiv:2401.07851.

Yu Xia, Fang Kong, Tong Yu, Liya Guo, Ryan A Rossi, Sungchul Kim, and Shuai Li. 2024c. Which llm to play? convergence-aware online model selection with time-increasing bandits. In Proceedings of the ACM on Web Conference 2024, pages 4059–4070.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453.

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.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2024. Multi-candidate speculative decoding. arXiv preprint arXiv:2401.06706.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Conference on Empirical Methods in Natural Language Processing (EMNLP).

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.

Ming Yin, Minshuo Chen, Kaixuan Huang, and Mengdi Wang. 2024. A theoretical perspective for speculative decoding algorithm. arXiv preprint arXiv:2411.00841.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. arXiv preprint arXiv:2309.12284.

Yao Zhao, Connor Stephens, Csaba Szepesvári, and Kwang-Sung Jun. 2023. Revisiting simple regret: Fast rates for returning a good arm. In International Conference on Machine Learning, pages 42110–42158. PMLR.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. Advances in Neural Information Processing Systems, 36.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation. arXiv preprint arXiv:2310.08461.

A Overview of appendix

This appendix provides supplementary material that expands on the main contents. Each section is designed to complement the research presented:

- **Appendix B**: Discusses the broader impact and further motivations of our work.
- **Appendix C**: Acknowledges the limitations of our current approach and outlines promising directions for future research.
- **Appendix D**: Provides a prepliminary for speculative sampling (SpS).
- **Appendix E**: Provides a comprehensive review of related work, situating our contributions within the broader context of speculative decoding with LLMs and multi-armed bandit research.
- **Appendix F**: Details additional experimental setups, offering further insights into the performance, behavior of our proposed method, and additional experimental results including long-context experiments, out-of-domain experiments, and evaluations with perturbed prompts.
- **Appendix G**: Presents rigorous mathematical proofs for the theoretical guarantees established in the main paper.
- **Appendix H**: Explores extensions to the MetaSD framework, addressing practical considerations such as switching costs and non-stationary environments.
- **Appendix I**: Offers further discussion and analysis of the results presented in the main paper, potentially including additional insights, interpretations, or comparisons.

Ethics statement This work primarily focuses on improving the efficiency of LLMs through algorithmic advancements and does not directly involve sensitive data or applications that could raise immediate ethical concerns.

Reproducibility statement To facilitate reproducibility, we provide a comprehensive exposition of the materials and experimental configurations within this paper and its accompanying appendices. The organization is as follows:

- **Section 2** - This section presents the problem statement and pseudocode for the MetaSD framework.
- **Section 3** & **Section H.3** - This section provide detailed MAB algorithms for the MetaSD framework under various scenarios.
- **Section 4** - This section elaborates on the implementation specifics, including the pre-trained models, datasets, and evaluation metrics.
- **Appendix F** - This section delves into additional details of the experimental settings.

The source code for this paper will be made publicly available under the MIT License upon publication.

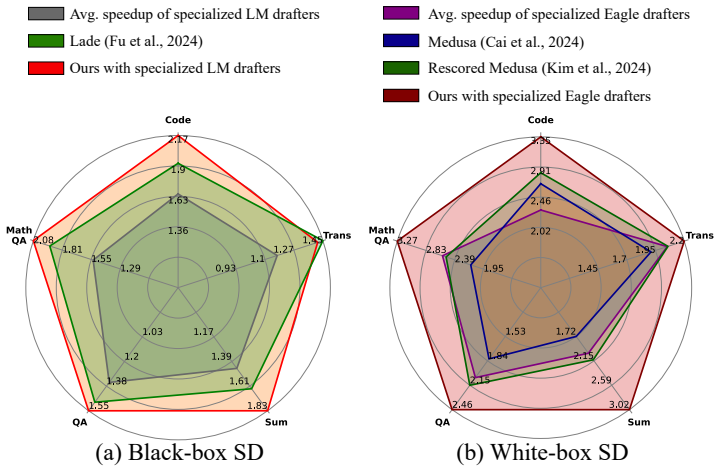
B Broader impact and further motivation

B.1 Broader impact

Generalized speedup Our MetaSD framework for multi-drafter speculative decoding has the potential to enhance the robust speedup capabilities of LLMs (Figure 4). By dynamically selecting from a diverse pool of drafters, the system can better adapt to a wider range of tasks and input contexts, potentially leading to reduced latency on unseen or less frequently encountered scenarios. This increased generalization could benefit various applications, such as machine translation, summarization, and creative writing, where models are often required to handle diverse and unpredictable inputs.

Efficiency The primary goal of our framework is to accelerate the inference process of LLMs. By leveraging speculative decoding with multiple drafters, we aim to achieve significant speedup gains compared to traditional single-drafter approaches. This improved efficiency could enable the deployment of large language models in resource-constrained environments or real-time applications where latency is critical. Faster inference could also facilitate broader accessibility to powerful language models, making them more practical for a wider range of users and use cases.

Systematic impact Our work remains various potential societal impact. Faster and more efficient language models could lead to advancements in various domains, such as healthcare, education, and customer service, where natural language understanding and generation play crucial roles.



(a) Black-box SD (b) White-box SD

Figure 4: Comparison of average speedup ratios by various methods relative to standard autoregressive greedy decoding on a single NVIDIA A100 GPU. The target model is Vicuna 7B v1.3. (a) Results for black-box methods. (b) Results for white-box methods. Detailed settings are in **Section 4**.

Table 9: Speedup ratio relative to the standard autoregressive greedy decoding on various multilingual datasets following (Yi et al., 2024) where target model is Vicuna 7B v1.3 and the drafter is decoder-only 68M language model: Japanese (Ja)→English (En) (Morishita et al., 2022), Russian (Ru)→En, German (De)→En (Bojar et al., 2016), French (Fr)→En (Bojar et al., 2014), and Chinese (Zh)→En (Barrault et al., 2019). Evaluations are conducted with a NVIDIA A5000 GPU.

Dataset	Ja-drafter	Ru-drafter	De-drafter	Fr-drafter	Zh-drafter
Ja →En	**1.757** ●	1.109	1.012	1.018	1.154
Ru →En	1.055	**1.817** ●	0.995	0.963	1.036
De →En	1.098	1.369	**2.360** ●	1.036	1.099
Fr →En	1.106	1.445	1.108	**2.135** ●	1.122
Zh →En	1.198	1.086	1.021	1.023	**1.516** ●

B.2 Further motivation

This subsection provides another line of research motivation in Section 2.1. MetaSD addresses the practical challenge of managing diverse and heterogeneous drafters often found in real-world systems (e.g., HuggingFace, Google Cloud, Azure, AWS, etc..). These drafters, pre-trained with varying objectives and frequently lacking detailed training documentation, pose significant obstacles to deployment frameworks that assume uniformity or rely on static selection strategies (e.g., rule-based strategies).

MetaSD provides a robust and adaptive mechanism for optimizing performance in environments characterized by task variability and drafter heterogeneity. By operating dynamically at the token level, it ensures task-specific efficacy without requiring retraining or fine-tuning of existing drafters. This flexibility allows MetaSD to excel in scenarios where traditional methods struggle, such as managing pre-trained drafters with black-box environment regarding the information for the use of drafters such as incomplete training histories or handling tasks with unpredictable distributions. Unlike frameworks that depend on rigid assumptions or predefined similarity metrics, MetaSD makes serving system particularly well-suited for organizations leveraging public repositories or heterogeneous resources.

Multilingual example In scenarios where the drafter's strengths do not align well with the task at hand, its predictions may be less accurate, leading to fewer accepted tokens and diminished speedup benefits of SD. As Table 9 shows, a drafter trained on a specific language pair exhibits significantly higher speedup on that pair compared to others, highlighting the need for a more adaptive approach. Therefore, integrating multiple heterogeneous drafters into the SD framework can potentially address this limitation.

C Limitation & future work

C.1 Limitation

Scalability It is important to acknowledge that the scalability of our approach may be challenged when dealing with an extremely large number of drafters. In such scenarios, the computational overhead associated with evaluating multiple drafters at each step could potentially outweigh the speedup benefits. To address this limitation, future work could explore strategies for pre-selecting a smaller subset of promising drafters based on initial query analysis or other heuristics, before applying the MetaSD framework. This would help to maintain the efficiency and scalability of our approach even in the presence of a vast pool of potential drafters.

Diverse target LLMs While our framework is designed to be agnostic to the target LLM architecture, extensive empirical evaluation across a wider range of LLMs is needed. Future work will assess the generalizability of our approach across different LLM architectures and sizes.

Batched inference Our current implementation primarily focuses on single-query scenarios. However, adapting the MetaSD framework to batched inference—where different tasks are mixed within a single batch—presents an opportunity for significant efficiency gains. Unlike static single-drafter-based SD, which can suffer from suboptimal performance when handling diverse tasks in a batch, MetaSD dynamically optimizes drafter selection at the instance level. This ensures consistently high throughput, even in high-throughput batched settings.

C.2 Future work

Reward design and exploration-exploitation balance The choice of reward function and the exploration-exploitation tradeoff significantly impact the performance of MetaSD. Exploring alternative reward designs and adaptive exploration strategies could lead to further improvements in speedup and adaptability.

Non-stationarity While we briefly discuss handling non-stationarity in Appendix H, more sophisticated techniques could be investigated. This could involve incorporating change detection mechanisms or developing MAB algorithms specifically tailored to the non-stationary nature of language generation.

Contextual bandits Our current framework primarily relies on observed rewards for drafter selection. Incorporating additional contextual information, such as the query type, user history, or drafter metadata, could lead to more informed decisions. Integrating contextual bandit algorithms into the MetaSD framework is a promising direction for future research.

Reinforcement learning (RL) formulation The MetaSD framework could also be formulated as an RL problem, where the agent learns to select the optimal drafter based on the current state (input context and generated text) to maximize a long-term reward (e.g., overall speedup). Exploring RL-based approaches could potentially uncover novel strategies for adaptive drafter selection.

MAB framework over different SD algorithms Our current work focuses on applying the MAB framework to select among heterogeneous drafters sharing the same SD algorithm (e.g., SpS or EAGLE). While this approach demonstrates significant benefits, it is worth noting that the MAB framework could potentially be extended to encompass a more diverse set of SD algorithms (e.g., Sps, PLD, Lookahead, EAGLE, and others). This would involve designing a reward function and selection strategy that can effectively compare and choose between fundamentally different SD approaches, each with its own strengths and weaknesses. Exploring this broader application of the MAB framework in speculative decoding is an interesting direction for future research.

D Preliminary: speculative sampling

Speculative decoding accelerates LLM inference by employing a smaller draft model to predict future tokens, which are then verified by the target LLM. This parallel token generation can significantly reduce latency, especially when the draft model's predictions align well with the target LLM's output distribution.

algorithm 3 outlines the speculative sampling procedure (Leviathan et al., 2023; Chen et al., 2023). Given an initial prompt sequence, the draft model generates E potential future tokens. Concurrently, the target LLM computes the probabilities of these tokens, as well as the probability of its own prediction for each subsequent token position. A drafted token is accepted if its probability, according to the target LLM, exceeds a certain threshold.

Algorithm 3: Speculative sampling (SpS)

INPUT : Target LLM $\mathcal{M}_p$, a small drafter $\mathcal{M}_q$, initial prompt sequence $x_1, \ldots, x_l$ and target sequence length B.

while $l < B$ **do**
  for $e \leftarrow 1, \ldots, E$ **do**
    $x_{l_e} \sim \mathcal{M}_q(x|x_1, \ldots, x_l, x_{l_1}, \ldots, x_{l_{e-1}})$
  end for
  In parallel, compute $E+1$ sets of logits drafts $x_{l_1}, \ldots, x_{l_E}$ with the target LLM $\mathcal{M}_p$:
    $\mathcal{M}_p(x|x_1, \ldots, x_l), \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l_1}), \ldots, \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l_1}, \ldots, x_{l_E})$
  for $j \leftarrow 1, \ldots, E$ **do**
    Sample $r \sim U[0, 1]$ from a uniform distribution
    if $r < \min(1, \frac{\mathcal{M}_p(x|x_1,\ldots,x_{l+j-1})}{\mathcal{M}_q(x|x_1,\ldots,x_{l+j-1})})$ **then**
      Set $x_{l+j} \leftarrow x_{l_j}$ and $l \leftarrow l+1$
    else
      Sample $x_{l+j} \sim (\mathcal{M}_p(x|x_1, \ldots, x_{l+j-1}) - \mathcal{M}_q(x|x_1, \ldots, x_{l+j-1}))_+$ and exit for loop.
    end if
  end for
  If all tokens $x_{l+1}, \ldots, x_{l+E}$ are accepted, sample extra token $x_{l+E+1} \sim \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l+E})$ and set $l \leftarrow l+1$
end while

This threshold is determined by comparing the target LLM's probability for the drafted token to both the draft model's prediction and a random sample, ensuring only high-confidence drafts are accepted. If a drafted token is rejected, the target LLM samples a token from the residual distribution, which represents the difference between its own prediction and the draft model's. This process iterates until the desired sequence length is reached.

Speculative sampling allows the target LLM to process multiple tokens in parallel by drafting them in advance, reducing the overall generation time. When the draft model's predictions are accurate, a significant portion of the generated tokens are accepted, leading to substantial speedup. The verification step and residual sampling ensure that the final generated sequence remains consistent with the target LLM's distribution, preserving generation quality. Speculative sampling provides a foundation for our proposed framework, where we extend this approach to incorporate multiple drafters and dynamically select the optimal one using MAB algorithms.

E Related work

E.1 Speculative decoding

Speculative decoding employs a draft-then-verify paradigm to enhance LLM inference speed. This approach tackles the latency bottleneck in autoregressive decoding, where extensive memory transfers for each token generation lead to underutilized compute resources (Patterson, 2004). Pioneering works by (Leviathan et al., 2023; Chen et al., 2023) introduced speculative decoding and sampling, enabling lossless acceleration of diverse sampling methods. These methods leverage smaller models within the same model family (e.g., T5-small for T5-XXL) without additional training. Models like Eagle (Li et al., 2024) and Medusa (Cai et al., 2024) integrate lightweight feedforward neural network heads into the LLM architecture, enabling early drafting of token sequences and improving throughput. Recent advancements have further refined speculative decoding by adopting judge model framework for accepting the draft tokens (Zheng et al., 2024) or combining with reward guided decoding in reasoning tasks (Liao et al., 2025).

Despite their efficacy, these methods often rely on a single drafter or a fixed set, limiting adaptability to diverse tasks and input contexts. Yi et al. (2024) propose specialized drafters based on the self-distilled dataset training, but dynamically selecting among heterogeneous drafters remains an open challenge. Liu et al. (2023) suggest online training of specialized drafters, but their reliance on query-based classification and limited speedup gains highlight the need for a more comprehensive

solution.

E.2 Bandit algorithms

Multi-armed bandit Multi-armed bandit (MAB) problem has been extensively studied for decades with various settings. For stochastic MAB setting, Lai and Robbins (1985) and Agrawal (1995) provided asymptotic optimal regret bounds that is logarithmic to the total round T and Auer (2002); Audibert et al. (2007) and Honda and Takemura (2010) proved this result also holds when T is finite. For another variant, EXP3 algorithm (Auer et al., 2002) proves the optimal regret bound in adversarial environment where reward distribution of each arm can change by adversary in every round.

Budgeted bandit The budgeted MAB problem address a bandit scenario where each arm pull yields both a reward and a cost drawn from individual distributions. Here, the goal is to maximize the cumulative reward until sum of the cost reaches the budget. Then, the optimal arm would be the one with the highest reward-to-cost ratio. ϵ-First policies (Tran-Thanh et al., 2010) and KUBE (Tran-Thanh et al., 2012) assumed a non-stochastic fixed cost for each arm pull. Ding et al. (2013) provided UCB-BV algorithm where cost for each arm is assumed to be a bounded discrete random variable.

Bandits with switching costs In real-world scenarios, a cost may be incurred whenever switching arms. This is related to the MAB problem with switching costs. (Dekel et al., 2014; Gao et al., 2019; Rouyer et al., 2021; Esfandiari et al., 2021; Amir et al., 2022). For stochastic MAB, Gao et al. (2019) and Esfandiari et al. (2021) assume a fixed cost is incurred whenever switching arms. They proved an instance-dependent regret bound $O(\log T)$ which does not depend on the unit switching cost value.

Pure exploration Pure exploration or best arm identification (BAI) problems (Even-Dar et al., 2002, 2006; Audibert and Bubeck, 2010) aim to explore as much as possible throughout the round to obtain the best arm at the end of the round. This contrasts with the traditional MAB objective which is maximizing cumulative reward. Even-Dar et al. (2002); Mannor and Tsitsiklis (2004); Even-Dar et al. (2006) investigated pure exploration in MAB under the PAC learning framework. BAI problems are primarily categorized into two settings. First, in the fixed budget setting (Audibert and Bubeck, 2010; Karnin et al., 2013; Carpentier and Locatelli, 2016), the goal is to minimize the chance of selecting sub-optimal arms within a fixed number of rounds. The other problem targets fixed confidence setting (Karnin et al., 2013; Jamieson et al., 2014; Garivier and Kaufmann, 2016; Chen et al., 2017) whose objective is to minimize number of rounds required to achieve a desired confidence level.

Non-stationary bandit Non-stationary bandit problems assume that reward distribution of each arm changes over time. The goal in non-stationary bandit problems is to find a balance between exploration and exploitation while carefully managing past information to adapt to the dynamic environment. Among the earliest works, Gittins (1974) assumed that only the best arm changes over time. This assumption was later relaxed in Whittle (1988), where the authors allow the mean reward for each arm to change at every round. Slivkins and Upfal (2008) assumed reward distribution follows a Brownian motion and established a regret upper bound that grows linear in rounds. Another line of works quantifies the degree of non-stationarity in the bandit instance by assuming a fixed value of L which represents a number of times reward distributions change. Auer et al. (2002) suggested EXP3.S algorithm and proved regret upper bound with given L but slightly worse when L is not given. (Kocsis and Szepesvári, 2006) suggested Discounted-UCB, where they obtain reward estimates with discounting factor over time. Garivier and Moulines (2011) introduced Sliding-window UCB, where they used fixed-size window to retain information of the rounds within the window for estimating mean reward. ADSWITCH in Auer et al. (2019) is proven to be nearly minimax optimal, achieving the state-of-the art regret bound without any prior knowledge of L.

E.3 Large language models and bandits

Recently, several works have made connections between LLMs with bandits using the emergent abilities of LLMs. One side of works utilize LLM as an agent to solve decision making problems combining with bandit framework (Baheri and Alm, 2023; Felicioni et al., 2024; Xia et al., 2024a; Park et al., 2024; Ong et al., 2024). Several works use bandit algorithms for improve the performance guarantee of LLMs with certain tasks such as for efficient prompt optimization (Shi et al., 2024) and online model selection (Xia et al., 2024c). The latter con-

cept has been further developed into LLM routing (Hu et al., 2024; Panda et al., 2025; Jitkrittum et al., 2025).

Concurrent works have leveraged the idea of bandit framework for SD. Liu et al. (2024) used Thomson sampling algorithm (which is one of the most popular bandit algorithm) to adaptively choose maximum candidate length N_{max} combining with early-exit framework. Huang et al. (2024) assume existence of multiple drafters and formulate SD as a contextual bandit problem. However, they rely on collecting offline samples for the policy learning which can be costly. Furthermore, their approach is regarded as a classification problem that the selected drafter is fixed in a single query.

Most relevant to ours, Hou et al. (2025) propose BANDITSPEC, which also leverage bandit algorithms for SD with multi-drafter scenario. While Hou et al. (2025) reported similar observations to ours but our work having key differences from them. First, while Hou et al. (2025) modify the confidence range of the UCB algorithm to control the regret analysis estimating mean accepted tokens, our theoretical analysis is built upon the budgeted bandit setting (Heyden et al., 2024) **which can be applied to more general reward shaping scenarios.** Moreover, our novel analysis on reward design shows that using estimation of mean acceptance rate as a reward (BD reward) which is obtained from the fine-grained model logits (not just the number of accepted tokens), improves the performance of the speculative decoding both practically and theoretically. Finally, Theorem 2 holds for any token generation instance (i.e, any realization of verified token instances) which is stronger than the expected regret over all token generation instances as done in Hou et al. (2025).

F Experiment detail

Since our experiment dataset covers broad and large dataset, we conduct single run experiment for the result unless specified otherwise.

F.1 Training specialized drafters with self-distilled data

Following the Yi et al. (2024), we use their training strategy consisting of two steps:

1. Pretraining drafters on a portion of C4 dataset (Raffel et al., 2019) and ShareGPT dataset (ShareGPT, 2023).
2. Finetuning the models with self distilled data having the target task with templates.

Self-distilled data Following prior work (Kim and Rush, 2016; Zhou et al., 2023; Cai et al., 2024; Yi et al., 2024), we generate the training data for specialized drafters through self-distillation from the target LLM. To capture the full spectrum of its output variability, we generate multiple responses at various temperatures—{0.0, 0.3, 0.7, 1.0}. We utilize this self-distilled dataset for training both independent small drafter models and dependent Eagle drafters. For Eagle-specific training details, we adhere to the settings outlined in the original Eagle paper (Li et al., 2024).

F.2 Drafter details

All independent drafters are based on a decoder-only Llama transformer model with 68M parameters. The model configuration includes 2 hidden layers, 768 hidden size, 12 attention heads, and a vocabulary size of 32,000. Other key settings are: silu activation function, 0.0 attention dropout, and no weight decay. The training recipe involves pretraining on a subset of the C4 and ShareGPT datasets, followed by fine-tuning on task-specific data generated through self-distillation from the target LLM. We employ 4 NVIDIA A100 GPUs with 80GB memory, utilizing techniques like FSDP (Fully Sharded Data Parallelism), gradient checkpointing, and lazy preprocessing to optimize training efficiency. Hyperparameters include a batch size of 8, 3 training epochs, a learning rate of 2e-5, and a cosine learning rate scheduler with a warmup ratio of 0.03. We maintain consistent architecture and training procedures across all white-box drafters, ensuring their heterogeneity stems solely from the diverse task-specific datasets they are fine-tuned on. For further specifics on Eagle drafter training, we refer readers to the original Eagle paper (Li et al., 2024).

F.3 Datasets

All of the datasets for our experiments are used within the range of the intended use for the original paper, not

Training dataset We utilize a diverse collection of datasets to train our specialized drafters, ensuring their proficiency across various tasks and languages:

- ShareGPT (ShareGPT, 2023): A dataset of approximately 58,000 conversations scraped.

These conversations include both user prompts and responses from OpenAI's ChatGPT.

- WMT16 De→En (Bojar et al., 2016): A dataset for German-to-English machine translation, providing high-quality parallel text data.
- JparaCrawl-v3.0 (Morishita et al., 2022): A large-scale Japanese web corpus, enabling training of a drafter specialized in Japanese-to-English translation.
- WMT16 Ru→En (Bojar et al., 2016): A parallel corpus for Russian-to-English machine translation, similar to the WMT16 De→En dataset but focusing on the Russian language.
- WMT14 Fr→En (Bojar et al., 2014): A dataset for French-to-English machine translation, providing additional multilingual training data.
- WMT19 Zh→En (Barrault et al., 2019): A dataset for Chinese-to-English machine translation, further expanding the language coverage of our drafter pool.
- Code alpaca (Chaudhary, 2023): A dataset of code generation instructions and corresponding outputs, facilitating the training of a drafter specialized in code-related tasks.
- CNN/Daily mail (Hermann et al., 2015): A dataset for summarization, comprising news articles and their corresponding summaries.
- Natural question answering (Kwiatkowski et al., 2019): A large-scale question answering dataset based on real user queries and Wikipedia passages, aiding in training a drafter for question answering tasks.
- Meta math question answering (Yu et al., 2023): A dataset focusing on mathematical question answering, providing specialized training data for a math-oriented drafter.

Evaluation dataset

- Multilingual translation: Ja to En (Morishita et al., 2022), Ru to En, De to En (Bojar et al., 2016), Fr to En (Bojar et al., 2014), and Zh to En (Barrault et al., 2019).
- Code generation: Code tasks from the MT-Bench dataset (Zheng et al., 2024).
- Summarization: CNN/Daily summarization dataset (Hermann et al., 2015).
- Question answering: Natural Questions dataset (Kwiatkowski et al., 2019).
- Math reasoning: GSM8K mathematical reasoning dataset (Cobbe et al., 2021).

Templates We employ specific prompt templates during model evaluation to guide the behavior of the target LLM and drafters, ensuring consistency and clarity in task execution. These templates are carefully designed to elicit desired responses and provide relevant context for each task category. Before the data templates, system prompts of LLMs are positioned at the front to provide additional context or instructions.

- Multilingual translation: 'Translate this sentence from [source language] to English: [source sentence]'.
- Code generation: Its instruction depends on the query.
- Summarization: 'Summarize: [article text]'.

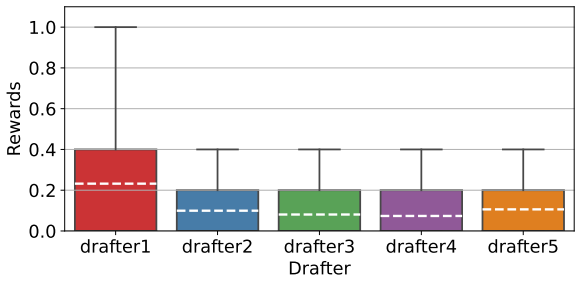
F.4 MAB settings

In our experiments, we set the exploration strength β for MetaSD-UCB to 0.01, balancing exploration and exploitation. For MetaSD-EXP3, we use a gamma value of 0.4 to control the degree of exploration. In the SH algorithm, we set the period to 1, ensuring frequent elimination of underperforming drafters.

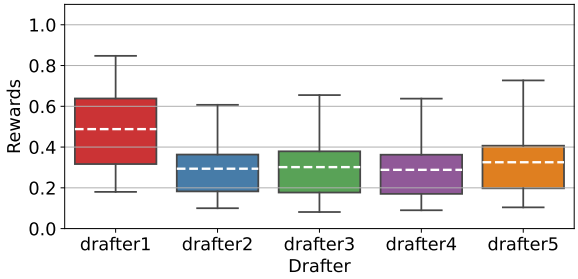
F.5 Baseline

We conduct several SD methods, ensuring their open-source availability and robust performance. Each method embodies a distinct strategy for accelerating LLM inference:

- SpS (Chen et al., 2023): SpS employs a smaller LM from the same model series as the drafter. In the verification stage, if a token is rejected, SpS corrects it using residual probability to maintain generation quality.
- BPD, Medusa, and Eagle (Stern et al., 2018; Cai et al., 2024; Li et al., 2024): These methods enhance the target LLM by incorporating additional lightweight FFN heads. These



(a) BE reward



(b) BD reward

Figure 5: Comparison of rewards on the Ja→En dataset across different drafters in two scenarios: (a) BE and (b) BD. Box plots show the distribution of rewards, with whiskers extending to the 5th and 95th percentiles. Drafter specializations: 1: Ja →En, 2: Ru →En, 3: De →En, 4: Fr →En, 5: Zh →En.

heads draft potential token sequences based on the penultimate layer representations from the target LLM.

- PLD (Saxena, 2023): Implementing the ideas of (Yang et al., 2023), PLD selects text spans directly from the input to serve as drafts, aiming for relevant and accurate initial predictions.

- R-BPD (Rescored blockwise parallel decoding) and R-Medusa (Rescored Medusa) (Kim et al., 2024): This method enhances BPD by rescoring the drafts at test-time, aiming to increase the number of accepted tokens.

F.6 Reward distribution

Figure 5 and Table 2 present a statistical analysis of the BE and BD reward distributions, collected using autoregressive decoding with the same Japanese dataset and drafter configurations as in Table 9. Several key observations emerge:

- Lower variance: The BD reward exhibits lower variance compared to the BE reward across all drafters. This suggests that BD provides a more stable and consistent feedback signal, leading to faster convergence with less sample complexity.

- Improved discrimination: The difference in mean reward between the optimal drafter (Drafter 1; Ja-drafter) and the suboptimal drafters is more pronounced with the BD reward. This improved discrimination between drafters can facilitate quicker identification of the optimal drafter by the MAB algorithm.

- Reduced sparsity: A significant portion of the BE rewards are zero, particularly for the suboptimal drafters. This sparsity can hinder the learning process of the MAB algorithm. In contrast, the BD reward consistently provides non-zero feedback, enabling continuous learning and adaptation.

These observations collectively suggest that the BD reward offers several advantages over the BE reward in the context of MetaSD. Its lower variance, improved discrimination between drafters, and reduced sparsity contribute to a more informative and efficient learning signal for the MAB algorithm, potentially leading to faster convergence and better overall performance.

For the experiment, we use the same Japanese dataset and drafter configurations as in Table 9, employing autoregressive decoding to collect BE and BD rewards at each step without actual speculative execution.

F.7 Long-context De →En translation

While our results in Table 3 and Table 5 have the relatively less effectiveness of MetaSpS on the WMT16 De→En translation task than other tasks, it is worth noting that this dataset primarily consists of relatively short sentences with an average length of fewer than 100 tokens. To assess the performance of our framework in a more challenging long-context scenario, we evaluate it on a new De→En translation dataset with an average context length of 500 tokens generated by GPT-4o. As shown in Table 10, MetaSpS-UCB achieves a speedup ratio of 2.031 on this long-context dataset, approaching the performance of the optimal drafter (Drafter3).

F.8 Evaluations on out-of-domain datasets

To evaluate the adaptability and performance of our MetaSD framework in out-of-domain settings, we

Table 10: Speedup ratio on long-context De→En translation with the same settings in Table 5.

Dataset	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	UCB
Long De→En	1.238	1.316	2.044	0.970	1.187	2.031

conduct additional experiments using the Alpaca-Finance (Bhartia, 2023) and RAG datasets (Xia et al., 2024b). These datasets fall outside the domains of the specialized drafters used in our main experiments, providing a robust test of MetaSD's ability to generalize. The results in Table 11, measured using an NVIDIA A100 GPU, are presented.

Superior adaptability The results indicate that MetaSD consistently outperforms both OFA drafters and most of individual specialized drafters in out-of-domain scenarios. This highlights its ability to dynamically adapt to new tasks without relying on prior assumptions about domain similarity. The following provides the limitations of similarity-based selection:

- Computing similarity between sentence embeddings requires encoding the context to generate embeddings. For inputs exceeding 128 tokens, this process can significantly increase inference time. For example, with over 100 tokens, similarity computation becomes slower than MetaSD's dynamic drafter selection.
- High accuracy in selecting the correct drafter based on embeddings is challenging, leading to potential misclassifications. Errors in this step can result in suboptimal drafter performance. For example, as input lengths increase, the performance gap between static Math drafters and MetaSD-UCB narrows, reducing the benefits of static drafter selection.

Intractability with heterogeneous drafters In practical scenarios, heterogeneous drafters often lack complete or uniform training descriptions. Under such conditions, similarity-based selection becomes infeasible. MetaSD's dynamic and adaptive approach offers a scalable alternative, ensuring robust performance even with limited information about drafter specialization.

F.9 Additional evaluations with perturbed prompts

In addition to the results in Section 4, Table 12 also shows the strength of MetaSD's dynamic token-level selection mechanism, which adapts to the token distributions during inference rather than relying solely on the characteristics of the input prompt.

F.10 Throughput over Eagle drafters

To evaluate the throughput efficiency of our proposed method, particularly in distributed system deployments where batch processing plays a critical role, we conduct experiments under the same settings described in the original Eagle paper (Li et al., 2024). Using an RTX 3090 (24GB) with the Vicuna 7B model, we measured throughput across a diverse set of tasks. The results demonstrate that MetaEagle-UCB achieves superior throughput compared to the single OFA Eagle, with a speedup factor of **2.427** versus **2.235** for single drafters.

A key strength of our drafter management mechanism lies in its ability to maintain throughput efficiency comparable to single-drafter methods. This is facilitated by preloading drafter parameters into DRAM, thereby avoiding frequent memory transfers to VRAM during computation. As a result, both the number of memory movements and the overall memory bandwidth requirements remain consistent with those of single-drafter configurations, even in scenarios involving multiple drafters. Additionally, the computational structure of MetaSD is designed to scale effectively across batches. Performance gains observed in single-batch scenarios carry over seamlessly to multi-batch settings, ensuring throughput efficiency in real-world distributed environments.

F.11 MetaEagle-UCB with Efficient KV Cache Strategies

In our framework, the KV cache is recalculated for the previous context whenever a drafter switch occurs. Despite this recalculation, the computational overhead is negligible, even for relatively long contexts. This efficiency arises from the minimal cost of prefilling the KV cache for a small drafter. For instance, in the Eagle drafter, only one layer of KV cache is computed for the unseen context, ensuring computational efficiency.

To further validate the framework's efficiency, we conducted additional experiments incorporat-

Table 11: Performance of MetaSpS, MetaEagle, and baselines on out-of-domain datasets (measured on A100 GPU).

Dataset	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	OFA Drafter	MetaSpS-UCB	EAGLE1	EAGLE2	EAGLE3	EAGLE4	EAGLE5	OFA Eagle	MetaEagle-UCB
RAG	1.720	1.373	1.752	1.944	1.552	1.638	**1.799**	1.844	1.568	2.566	2.535	1.793	2.175	**2.238**
Finance	1.416	1.284	1.414	1.550	1.397	1.367	**1.436**	2.432	2.175	2.494	2.826	2.175	2.435	**2.517**

Table 12: Black-box performance with perturbed prompts (speedup relative to greedy decoding, measured on A100 GPU).

Task	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	OFA Drafter	MetaSpS - UCB
Code	2.368	1.158	1.521	1.763	1.633	1.937	**2.139**
Translation	0.997	1.986	0.973	1.036	0.935	0.969	**1.422**
CNN	1.458	1.016	1.895	1.458	1.318	1.521	**1.779**
NQA	1.297	1.158	1.285	1.907	1.237	1.387	**1.610**
MathQA	1.482	1.184	1.357	1.470	2.346	1.895	**2.149**

ing StreamingLLM techniques (Xiao et al., 2023). These techniques circumvent the need for full KV cache recalculation, offering an alternative method for reducing computational costs. The results, summarized in Table 13, demonstrate that StreamingLLM achieves comparable performance to the default approach of KV cache recalculation, highlighting the robustness of MetaSD.

These results confirm two key observations. First, the computational overhead introduced by full KV cache recalculation is minimal, as evidenced by MetaEagle-UCB maintaining high performance across tasks. This demonstrates that recalculating the KV cache is not a significant bottleneck. Second, Streaming Decode techniques provide an effective alternative, yielding similar overall performance with slight improvements observed in specific cases such as Translation and QA. These findings underscore the flexibility and efficiency of MetaSD in managing KV cache strategies.

F.12 Temperature sampling

We investigate the impact of temperature sampling on MetaSpS performance. Table 14 presents the speedup ratios achieved with temperature sampling with temperature 0.7 on an NVIDIA A6000 GPU. Consistent with the trends observed in our main experiments with greedy decoding, MetaSD continues to achieve competitive speedup.

To further assess the robustness of MetaSD under stochastic decoding conditions, we conduct experiments with temperature sampling ($T = 1.0$). The results, presented in Table 15 and Table 16, align with our theoretical assumptions, demonstrating the adaptability of MetaSD even in non-deterministic decoding settings. These results further support the theoretical alignment of MetaSD with stochastic decoding conditions.

F.13 Throughput evaluation in single-batch and heterogeneous settings

We evaluate its throughput performance in both single-batch and heterogeneous batch settings. As shown in Table 17, we conduct experiments on out-of-distribution tasks—Physics QA (Li et al., 2023), Hotpot QA (Yang et al., 2018), and MMLU-CoT (Hendrycks et al., 2021b,a)(average across 57 tasks). These datasets provide a robust evaluation of MetaSD's ability to generalize beyond training domains.

In single-batch settings, MetaEagle-UCB consistently outperforms OFA Eagle, achieving higher throughput across all evaluated tasks. This improvement stems from its adaptive selection mechanism, which efficiently routes queries to the most suitable drafter, even in OOD scenarios.

For heterogeneous batch throughput, where different tasks are mixed within a batch, MetaEagle-UCB remains competitive. While throughput slightly decreases in heterogeneous settings due to increased I/O from drafter switching, its performance remains comparable to single-drafter methods in small-batch inference. These findings highlight MetaSD's ability to balance adaptive specialization with high-throughput efficiency, making it well-suited for real-world deployment in diverse task distributions.

F.14 Statistical verification

All reported values in our experiments are averaged over three independent runs to support statistical reliability.

Table 13: Performance comparison of MetaSD-UCB with different KV cache strategies (speedup relative to standard greedy decoding, measured on A100 GPU).

Task	MetaEagle-UCB (Recomputing KV)	MetaEagle-UCB with StreamingLLM
Code	3.724	3.624
Trans	2.318	2.352
Sum	3.057	2.986
NQA	2.641	2.654
Math	3.520	3.338

Table 14: Speedup ratio with temperature sampling as temperature is set to 0.7 over a NVIDIA A6000 GPU.

Dataset	SpS with specialized drafters					Bandit
	Drafter1	**Drafter2**	**Drafter3**	**Drafter4**	**Drafter5**	**UCB**
Code	**2.250**	1.215	1.379	1.532	1.513	1.896
Trans	1.086	**1.886**	1.096	1.130	1.078	1.431
Sum	1.461	1.165	**1.874**	1.463	1.353	1.744
QA	1.316	1.193	1.324	**1.776**	1.272	1.534
Math	1.450	1.258	1.355	1.616	**2.379**	2.046

Table 15: Performance of MetaSpS under stochastic decoding (temperature = 1.0). Speedup ratios are reported relative to standard autoregressive greedy decoding.

Task	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	OFA	MetaSpS-UCB
Code	1.781	0.963	1.168	1.260	1.178	1.501	**1.596**
Translation	0.856	1.695	0.897	0.880	0.838	0.861	**1.197**
CNN	1.201	0.918	1.629	1.223	1.092	1.230	**1.439**
NQA	1.073	0.961	1.132	1.510	1.031	1.123	**1.322**
MathQA	1.220	1.026	1.200	1.360	1.968	1.512	**1.673**

Table 16: Performance of MetaEagle under stochastic decoding (temperature = 1.0). Speedup ratios are reported relative to standard autoregressive greedy decoding.

Task	EAGLE1	EAGLE2	EAGLE3	EAGLE4	EAGLE5	OFA Eagle	MetaEagle-UCB
Code	3.019	0.872	1.012	1.348	1.809	2.926	**2.765**
Translation	1.030	1.817	1.373	1.278	0.997	1.578	**1.668**
CNN	0.998	0.834	2.289	1.267	0.864	1.749	**1.935**
NQA	1.179	0.922	1.269	2.181	1.011	1.680	**1.756**
MathQA	1.739	0.966	1.462	2.141	3.099	2.289	**2.539**

Table 17: Throughput in single-batch and heterogeneous batch settings.

Task / Batch Size	Single Batch Throughput		Heterogeneous Batch Throughput		
	OFA Eagle	**MetaEagle-UCB**	**Single OFA Eagle**	**Homogeneous (MetaEagle-UCB)**	**Heterogeneous (MetaEagle-UCB)**
Physics	2.424	2.573	—	—	—
Hotpot QA	2.262	2.270	—	—	—
MMLU-CoT (Avg. 57 tasks)	2.466	2.529	—	—	—
Batch Size = 1	—	—	2.803	3.045	3.045
Batch Size = 2	—	—	2.751	2.933	2.518
Batch Size = 4	—	—	2.563	2.701	1.931

Table 18: Mathematical terms and notations in our work.

Notation	Descriptions
K	Number of drafters
$[K]$	For a given integer K, denotes the set $\{1, ..., K\}$
i	Drafter index $i \in [K]$
α_i	True mean of acceptance rate when using drafter i
$i^\star$	Drafter index with the highest α_i
t	Number of current round
B	Total number of tokens to generate
$l(t)$	Number of input tokens at round t
$x^{1:l}$	Token sequence of first l tokens
$\mathcal{M}_q$	Target model
$\mathcal{M}_{q_i}$	The i-th drafter
p^l	Probability distribution of target model output given token sequence $x^{1:l}$
q_i^l	Probability distribution of output of drafter i given token sequence $x^{1:l}$
$N_{acc}(i,t)$	Number of accepted tokens using drafter i in round t
N_{max}	Number of candidate tokens
r	Arbitrary reward distribution with bounded support [0,1]
$r_{i,t}$	General reward feedback using drafter i in round t
$r_{i,t}^{BE}$	BE reward using drafter i in round t
$r_{i,t}^{BD}$	BD reward using drafter i in round t
$n_i(t)$	Number of selecting drafter i until round t
a_t	Index of selected drafter in round t
β	The exploration strength hyperparamter in UCB
γ	The exploration hyperparameter used in EXP3
μ_i	Expectation of the reward distribution of drafter i
π	Bandit policy (algorithm)
$\tau(\pi, B)$	Stopping time for the policy π with given total number of tokens B
λ	Switching cost constant factor
Δ_i	Suboptimality gap for the arbitrary reward distribution r: $\mu_i^\star - \mu_i$
$\Delta(\alpha_i)$	Suboptimality gap for the BD reward: $\alpha_i^\star - \alpha_i$
Δ_i^{BE}	Suboptimality gap for the BE reward
$R(r_i)$	Feedback signal for reward distribution when using drafter i (Theorem 1)
$d_{TV}(\cdot,\cdot)$	The total variation distance between probability measures
$\mathbb{I}$	Indicator function
$O(\cdot)$	Big O notation

G Proofs

To begin, we provide the mathematical terms and notations in Table 18.

G.1 Basic lemmas

First, we provide basic concentration inequalities which will be used to prove our theoretical results.

Lemma 3 (Chernoff-Hoeffding bound)**.** *Suppose there are n random variables $X_1, X_2, \ldots, X_n$ whose value is bounded in $[0,1]$ and $\mathbb{E}[X_t|X_1, \ldots, X_{t-1}] = \mu$ for $2 \le t \le n$. Then, for $S_n = \sum_{i=1}^{n} X_i$ and $a \ge 0$, following inequalities holds:*

$$\mathbb{P}(S_n \ge n\mu + a) \le e^{-2a^2/n},$$
$$\mathbb{P}(S_n \le n\mu - a) \le e^{-2a^2/n}.$$

Lemma 4 (Bernstein inequality)**.** *Suppose there are n random variables $X_1, X_2, \ldots, X_n$ whose value is bounded in $[0,1]$ and $\sum_{t=1}^{n} \mathrm{Var}[X_t|X_{t-1}, \ldots, X_1] = \sigma^2$. Then, for $S_n = \sum_{i=1}^{n} X_i$ and $t \ge 0$, following inequalities holds:*

$$\mathbb{P}(S_n \ge \mathbb{E}[S_n] + t) \le \exp(-\frac{t^2}{\sigma^2 + t/2}).$$

G.2 Proof of Theorem 1

In order to prove the theorem, we first provide statistics for the BE and BD rewards by the following lemmas.

BE reward statistics Here, we explicitly calculate expectation and variance of the BE reward in one round of speculative decoding. The result is presented in the following lemma.

Lemma 5 (BE reward statistics)**.** *The expectation and variance of the number of accepted tokens is as follows:*

$$\begin{aligned}
&\mathbb{E}[r_{i,t}^{BE}] = \frac{\alpha_i - \alpha_i^{N_{max}+1}}{N_{max}(1-\alpha_i)}, \\
&\mathrm{Var}[r_{i,t}^{BE}] = \frac{1}{(N_{max})^2(1-\alpha_i)^2} \\
&\cdot \alpha_i(1 - (2N_{max}+1)\alpha_i^{N_{max}} \\
&+ (2N_{max}+1)\alpha_i^{N_{max}+1} - \alpha_i^{2N_{max}+1}).
\end{aligned} \tag{8}$$

Proof of Lemma 5 We first start with calculating the expectation and variance of N_{acc} which can be obtained in a closed form. Suppose we conduct one round of speculative decoding for candidate token indices $l+j$ for $j = 1, \ldots, N_{max}$. Now, define E_{l+j}^i as the event of $(l+j)$-th token generated by drafter i is accepted in the verification stage. Also, define random variable X_{l+j}^i to be 1 when E_{l+j}^i occurs and 0 otherwise. With the stationary assumption, one can observe X_{l+j}^i follows Bernoulli distribution with mean α_i. Now, expectation can be obtained as:

$$\begin{aligned}
\mathbb{E}[N_{acc}(i,t)] &= \sum_{l=1}^{N_{max}} \mathbb{E}[X_{l+j}^i] \\
&= \sum_{l=1}^{N_{max}} \alpha_i^l = \frac{\alpha_i - \alpha_i^{N_{max}+1}}{1-\alpha_i}.
\end{aligned} \tag{9}$$

To obtain variance, from $X_{L+l}^i \sim Ber(\alpha_i^l)$, following holds:

$$\mathrm{Var}(X_{L+l}^i) = (\alpha_i^l - \alpha_i^{2l})$$

Now, we can directly obtain a closed form of the variance by,

$$\begin{aligned}
&\mathrm{Var}(N_{acc}(i,t)) = \mathrm{Var}(\sum_{l=1}^{N_{max}} X_{L+l}^i) \\
&= \sum_{l=1}^{N_{max}} \mathrm{Var}(X_{L+l}^i) + 2 \cdot \sum_{l<m} \mathrm{Cov}(X_{L+l}^i, X_{L+m}^i) \\
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \mathrm{Cov}(X_{L+l}^i, X_{L+m}^i) \\
&- \sum_{l=1}^{N_{max}} \mathrm{Var}(X_{L+l}^i) \\
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} (\alpha_i^m - \alpha_i^{m+l}) - \sum_{l=1}^{N_{max}} (\alpha_i^l - \alpha_i^{2l})
\end{aligned} \tag{10}$$

$$
\begin{aligned}
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \alpha_i^m - 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \alpha_i^{m+l} \\
&- \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2} \\
&= 2 \cdot \sum_{l=1}^{N_{max}} l \cdot \alpha_i^l \\
&- 2 \cdot \sum_{l=1}^{N_{max}} \alpha_i^l \left(\frac{\alpha_i^l - \alpha_i^{N_{max}+1}}{1-\alpha_i} \right) \\
&- \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2} \\
&= 2 \cdot \sum_{l=1}^{N_{max}} l \cdot \alpha_i^l - 2 \cdot \frac{1}{1-\alpha_i} \sum_{l=1}^{N_{max}} \alpha_i^{2l} \\
&+ 2 \cdot \frac{\alpha_i^{N_{max}+1}}{1-\alpha_i} \sum_{l=1}^{N_{max}} (\alpha_i^l \\
&- \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}) \\
&= \frac{2\alpha_i(N_{max} \cdot \alpha_i^{N_{max}+1} - (N_{max}+1)\alpha_i^{N_{max}} + 1)}{(1-\alpha_i)^2} \\
&- \frac{2\alpha_i^2(1-\alpha_i^{2N_{max}})}{(1-\alpha_i)(1-\alpha_i^2)} \\
&+ \frac{2\alpha_i^{N_{max}+2}(1-\alpha_i^{N_{max}})}{(1-\alpha_i)^2} \\
&- \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}.
\end{aligned}
$$

The second equality comes from the basic property of variance, the fourth equality is from observing $\mathrm{Cov}(X_{L+l}^i, X_{L+m}^i) = \mathbb{E}[X_{L+l}^i X_{L+m}^i] - \mathbb{E}[X_{L+l}^i]\mathbb{E}[X_{L+m}^i] = \alpha_i^m - \alpha_i^{l+m}$. After rearranging the terms, we can obtain closed form of the variance as follows.

$$
\begin{aligned}
&\mathrm{Var}(N_{acc}(i,t)) = \frac{\alpha_i}{(1-\alpha_i)^2} \cdot \\
&(1 - (2N_{max}+1)\alpha_i^{N_{max}} \\
&+ (2N_{max}+1)\alpha_i^{N_{max}+1} - \alpha_i^{2N_{max}+1})
\end{aligned} \tag{11}
$$

Since $r_{i,t}^{BE} = \frac{1}{N_{max}} N_{acc}(i,t)$ by definition, plugging this into eq. 9 and eq. 11 concludes the proof. □

BD reward statistics Next, we obtain the expectation and variance of the BD reward by following lemma.

Lemma 6. *Following the relationships hold for $r_{i,t}^{BD}$ for all i, t:*

$$
\mathbb{E}[r_{i,t}^{BD}] = \alpha_i, \mathrm{Var}[r_{i,t}^{BD}] \leq \frac{1}{4N_{max}} \tag{12}
$$

Proof of Lemma 6 Under stationary assumption, any random variable which is bounded in $[0,1]$ has variance less than $\frac{1}{4}$. Since in eq. 4, $r_{i,t}^{BD}$ is constructed by empirical mean of N_{max} numbers of samples under stationary assumption, following holds:

$$
\begin{aligned}
&\mathrm{Var}[r_{i,t}] \\
&= \mathrm{Var}\left[\frac{1}{N_{max}} \sum_{j=0}^{N_{max}-1} (1 - d_{TV}(p^{l(t)+j}, q_i^{l(t)+j}) \right] \\
&\leq \frac{1}{4N_{max}},
\end{aligned}
$$

and this concludes the proof. □

Relationship between expectations of two rewards. Combining above lemmas, one can show that the expectation of the BD reward is proportional to the BE reward.

Lemma 7. *Following relationship holds between the expectation of the BE reward and the expectation of the BD reward:*

$$
\mathbb{E}[r_{i,t}^{BE}] = \frac{1-\alpha_i^{N_{max}}}{N_{max}(1-\alpha_i)} \mathbb{E}[r_{i,t}^{BD}]. \tag{13}
$$

Proof. Combining Lemma 5 and Lemma 6 directly gives the result. □

Note that $\alpha_{i,t}$ can be interpreted as the acceptance rate for the t-th token generated by the i-th model (Leviathan et al., 2023).

Bandit feedback signal Next, we formally define the feedback signal with following definition.

Definition 3 (Feedback signal)**.** *Under stationary environment, any reward design r_i with $\mu_i = \mathbb{E}[r_i]$, $i^\star = \arg\max \mu_i$, and $\Delta_i = \mu_i^\star - \mu_i$, we define feedback signal for each suboptimal arm $i \neq i^\star$ as follows.*

$$
R(r_i) := \frac{\Delta_i^2}{\max(\mathrm{Var}[r_i], \mathrm{Var}[r_{i^\star}])}
$$

As we will see, R become crucial factor that governs regret upper bound of our MetaSD-UCB algorithm. Specifically, the lower $R(r_i)$ guarantees smaller amount of regret by picking suboptimal arm i.

Then, we provide a formal version of Theorem 1 which states the BD reward actually has lower feedback signal compared to the BE reward.

Theorem 3 (Formal version of Theorem 1)**.** *Denote $\Delta(\alpha_i) := \alpha_{i^\star} - \alpha_i$ for any suboptimal arm i and $n := N_{max}$ for notational convenience. For any $n \in \mathcal{N}$, define functions f_n, g_n, h_n on $(0,1)$ by $f_n(x) = \frac{x - x^{n+1}}{1-x}$, $g_n(x) = f_n'(x) = \sum_{s=1}^{n} s x^{s-1}$, and $h_n(x) = \sum_{s=1}^{n} s(x^{s-1} - x^{2n-s})$. Then following holds:*

$$R(r_i^{BD}) \geq 4(\Delta(\alpha_i))^2 N_{max}. \tag{14}$$

Also, following holds for any drafter configuration satisfying $h_n(\alpha_{i^\star}) \geq \frac{g_n(\alpha_{i^\star})^2}{4n\alpha_{i^\star}}$ and $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$:

$$R(r_i^{BD}) > R(r_i^{BE}). \tag{15}$$

Proof. Upper bound for the BD reward can be directly obtained from Lemma 6. To prove eq. 15, denote $N_{max} = n$ for notational convenience. Then, by directly applying Lemma 5, it is observed that

$$\begin{aligned}
\frac{1}{R(r_i^{BE})} &= \frac{\max(\mathrm{Var}[r_i^{BE}], \mathrm{Var}[r_{i^\star}^{BE}])}{\Delta_i^2} \\
&= \frac{\alpha_{i^\star}(1 - (2n+1)\alpha_{i^\star}^n + (2n+1)\alpha_{i^\star}^{n+1} - \alpha_{i^\star}^{2n+1})}{(f_n(\alpha_i^\star) - f_n(\alpha_i))^2 (1-\alpha_{i^\star})^2} \\
&> \frac{\alpha_{i^\star}(1 - (2n+1)\alpha_{i^\star}^n + (2n+1)\alpha_j^{n+1} - \alpha_{i^\star}^{2n+1})}{(g_n(\alpha_{i^\star})\Delta(\alpha_i))^2 (1-\alpha_{i^\star})^2} \\
&= \frac{\alpha_{i^\star} h_n(\alpha_{i^\star})}{(g_n(\alpha_{i^\star})\Delta(\alpha_i))^2} \\
&\geq \frac{1}{4(\Delta(\alpha_i))^2 N_{max}},
\end{aligned} \tag{16}$$

where the first inequality is from f_n is a convex function, the second equality comes from Lemma 5, and the last line comes from the assumption. □

Practical considerations While Theorem 3 provides a general scenario, the inequalities used in its derivation can be quite loose in certain cases. In practice, the BD reward often exhibits a significantly smaller feedback signal $R(r_i)$ than the BE reward. For example, consider the case where $N_{max} = 5$, which is the setting used in our main experiments. The condition $h_n(\alpha_{i^\star}) > \frac{g_n(\alpha_{i^\star})}{4n\alpha_{i^\star}}$ holds for $0.06 < \alpha_{i^\star} < 0.8$, which covers most of the practical range of $\alpha_{i^\star}$. This implies that, in many realistic scenarios, the BD reward leads to a substantially tighter regret bound compared to the BE reward, further supporting its effectiveness in the MetaSD framework. Moreover, assumption of $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$ covers most of the practical scenarios. As an example, if $n = 5$, $\mathrm{Var}[r_i^{BE}]$ is monotonically increasing until $\alpha_i = 0.815$. Consequently, for any drafter set with $\alpha_{i^\star} < 0.815$, $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$ holds for all suboptimal drafters.

G.3 Stopping time regret

In this subsection, we provide the equivalence relation between two objectives, maximizing the reward and minimizing the stopping time. First, we define the regret of MetaSD in terms of the stopping time. Denote $\tau(\pi, B)$ as the stopping time for any policy π with target sequence length B and $\pi^\star$ as the optimal policy. In Definition 1, stopping time regret of policy π with B is defined as:

$$\textsc{Reg}^s(\pi, B) = \mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, B)]. \tag{17}$$

Intuitively, minimizing $\textsc{Reg}^s(\pi, B)$ should guarantee optimal speedup since minimizing $\tau(\pi, B)$ implies minimizing the number of total SD round. The following lemma proves that our reward design is well aligned with such objective.

Lemma 8 (BE reward original regret)**.** *For any policy π with the target sequence length B, denote the original regret objective using the BE reward as*

$$\textsc{Reg}^{o,BE}(\pi, T) = \sum_{t=1}^{T} (\mathbb{E}[r_{i^\star}] - \mathbb{E}[r_{a_t}]). \tag{18}$$

Then, the following equation holds:

$$\textsc{Reg}^{o,BE}(\pi, T) = \frac{1}{N_{max}} \textsc{Reg}^s(\pi, B). \tag{19}$$

Consequently, minimizing the regret in terms of accepted tokens is equivalent to minimizing $\textsc{Reg}^{(s)}(\pi, B)$.

Proof. It is observed that

$$\begin{aligned}
B &= \sum_{t=1}^{\tau(B)} (N_{acc}(i,t) + 1) \\
&= \tau(B) + \sum_{t=1}^{\tau(B)} N_{acc}(i,t) \\
&= \tau(B) + N_{max} \sum_{t=1}^{\tau(B)} r_{a_t,t}.
\end{aligned}$$

which leads to,

$$\tau(\pi, B) - \tau(\pi^\star, B) = N_{max} \sum_{t=1}^{\tau(B)} (r_{a_t^\star,t} - r_{a_t,t}), \tag{20}$$

where $a_t^\star$ is the action from the optimal policy $\pi^\star$ in round t. By taking the expectation on both sides, we get the result. □

However, above result does not hold in every reward design as can be seen in the following Lemma.

Lemma 9 (BD reward original regret)**.** *For any policy π with the fixed target sequence length B, denote the original regret objective using the BE reward as* $\text{REG}^{o,BD}(\pi, T) = \sum_{t=1}^{T}(\mathbb{E}[r_{i^\star}] - \mathbb{E}[r_{a_t}])$ *. Then, there exists a bandit instance with the two different policies π_1, π_2 such that:*

$$\mathbb{E}[Reg^{o,BD}(\pi_1, B)] < \mathbb{E}[Reg^{o,BD}(\pi_2, B)],$$
$$\mathbb{E}[Reg^{s}(\pi_1, B)] > \mathbb{E}[Reg^{s}(\pi_2, B)]. \tag{21}$$

Proof. Suppose we have three drafters with $\alpha_1 = 0.1, \alpha_2 = 0.5, \alpha_3 = 0.8$ with $N_{max} = 2$. Consider π_1 as the deterministic policy where it picks the drafter 1 for the first round and pick the drafter 3 rest of the rounds. Also, π_2 be the policy which picks drafter 2 for the first two rounds and drafter 3 for the rest of the rounds. For the original regret objective, π_1 has expected regret of 0.7 while π_2 has expected regret 0.6. However, expectation of number of accepted tokens until first two rounds becomes $(0.1+0.1^2)+(0.8+0.8^2) = 1.55$ for π_1 and $2(0.5+0.5^2) = 1.50$ for π_2. Since policy for the rest of the rounds are the same, we can conclude that the expected stopping time of policy π_1 is less than the expected stopping time of the policy π_2. As a result, π_2 is better in terms of original regret objective and π_1 is better with stopping time regret objective. □

The above lemma indicates that if we take BD reward design with original bandit regret objective in (Lattimore and Szepesvári, 2020), the optimal bandit algorithm may not be optimal in MetaSD algorithm. This necessitates us to define a new regret objective in Definition 1 with proper reward design.

G.4 MetaSD-UCB with general reward

In this subsection, we provide a generic theorem which is stated as follows.

Theorem 4 (Generic regret upper bound)**.** *For any reward design r, denote $\mu_i = \mathbb{E}[r_{i,t}]$, $\Delta_i = \mu_{i^\star} - \mu_i$, and $i^\star = \arg\max \alpha_i$. If $i^\star = \arg\max \mu_i$, then there exists a constant $C', C''' > 0$ such that following bound holds:*

$$\text{REG}(\pi, B) < \sum_{i \neq i^\star} \frac{8}{\Delta_i^2}(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C') + C'''. \tag{22}$$

Above theorem holds for any reward design as long as the drafter with the maximum expected reward $\mathbb{E}[r_{i,t}]$ also has the highest acceptance rate α_i. Since both the BD and BE rewards satisfy this condition, Theorem 4 applies to both of the reward designs. The proof of Theorem 4 consists of two main parts. First, given total round, we can bound the expected number of selecting suboptimal arms using the same anlysis in (Auer, 2002). Next, we get the upper bound on expected stopping time of MetaSD-UCB algorithm.

Bounding suboptimal selection Given fixed stopping time, we can bound the expectation of number of selecting suboptimal arms as follows:

Lemma 10 (Theorem 1 from (Auer, 2002))**.** *Let $n_i(t)$ be the number of pulling sub-optimal drafter ($i \neq i^\star$) by the MetaSD-UCB until round t. Also, denote $\Delta_i := \mu_{i^\star}^r - \mu_i^r$ be the sub-optimal gap. Then, following inequality holds for $\beta = 1$:*

$$\mathbb{E}[n_i(\tau(B))|\tau(B)] \leq \frac{8 \ln \tau(B)}{\Delta_i^2} + 1 + \frac{\pi^2}{3}. \tag{23}$$

Proof of Lemma 10 For the analysis, we restate the proof in (Auer, 2002) for MetaSD-UCB algorithm with our notations. One can observe $n_i(\tau(B))$, the number of times drafter i is chosen for the one round of speculative decoding until the

end of generation, can be bounded as follows:

$$
\begin{aligned}
n_i(\tau(B)) &= 1 + \sum_{t=K+1}^{\tau(B)} \mathbb{I}[a_t = i] \\
&\le l + \sum_{t=K+1}^{\tau(B)} \mathbb{I}[a_t = i, n_i(t-1) \ge l] \\
&\le l + \sum_{t=K+1}^{\tau(B)} \mathbb{I}[\hat{\mu}_{i,t-1} + \sqrt{\frac{2\ln(t-1)}{n_i(t-1)}} \ge \hat{\mu}_{i^\star,t-1} \\
&+ \sqrt{\frac{2\ln(t-1)}{n_{i^\star}(t-1)}}, n_i(t-1) \ge l] \\
&\le l + \sum_{t=1}^{\tau(B)} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} \mathbb{I} \\
&\left[\hat{\mu}_{i,n_i} + \sqrt{\frac{2\ln(t-1)}{n_i}} \ge \hat{\mu}_{i^\star,s} + \sqrt{\frac{2\ln(t-1)}{s}}\right].
\end{aligned} \tag{24}
$$

Here, $\mathbb{I}$ is an indicator function and l is a positive integer. Now, one can see following holds:

$$
\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i,n_i} + \sqrt{\frac{2\ln t}{n_i}} \ge \hat{\mu}_{i^\star,s} + \sqrt{\frac{2\ln t}{s}}\right) \\
&\le \mathbb{P}\left(\hat{\mu}_{i^\star,s} \le \mu_{i^\star} - \sqrt{\frac{2\ln t}{s}}\right) \\
&+ \mathbb{P}\left(\hat{\mu}_{i.n_i} \ge \mu_i + \sqrt{\frac{2\ln t}{n_i}}\right) \\
&+ \mathbb{P}\left(\mu_{i^\star} < \mu_i + 2\cdot\sqrt{\frac{2\ln t}{n_i}}\right).
\end{aligned}
$$

First term and the second term in the above equation is bounded by Lemma 3 as:

$$
\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i^\star,s} \le \mu_{i^\star} - \sqrt{\frac{2\ln t}{s}}\right) \le \exp(-4\ln t) = t^{-4}, \\
&\mathbb{P}\left(\hat{\mu}_{i.n_i} \ge \mu_i + \sqrt{\frac{2\ln t}{n_i}}\right) \le \exp(-4\ln t) = t^{-4}.
\end{aligned} \tag{25}
$$

By choosing $l = \lceil \frac{8\ln\tau(B)}{\Delta_i^2} \rceil$, one can see that the last term is 0 since,

$$
2\cdot\sqrt{\frac{2\ln t}{n_i}} \le 2\cdot\sqrt{\frac{2\ln t}{(\frac{8\ln\tau(B)}{\Delta_i^2})}} \le \Delta_i. \tag{26}
$$

Finally, taking expectation of eq. 24 and put the above result, one can see that:

$$
\begin{aligned}
&\mathbb{E}[n_i(\tau(B))|\tau(B)] \\
&\le \lceil \frac{8\ln\tau(B)}{\Delta_i^2} \rceil + 2\sum_{t=1}^{\tau(B)} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} 2t^{-4} \\
&\le \lceil \frac{8\ln\tau(B)}{\Delta_i^2} \rceil + 2\sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} 2t^{-4} \\
&\le \frac{8\ln\tau(B)}{\Delta_i^2} + 1 + \frac{\pi^2}{3}.
\end{aligned} \tag{27}
$$

□

Bounding stopping time The overall structure of the proof in bounding the stopping time is based on the proof of Lemma 4 in (Ding et al., 2013) while we provide additional details that suits with our problem formulation. First, we obtain upper bound on stopping time by following lemma:

Lemma 11. *Following inequalities holds for some constants $C', C'' > 0$:*

$$
\begin{aligned}
&\mathbb{E}[\tau(\pi, B)] \le \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ \sum_{i\ne i^\star} \frac{8}{\Delta_i^2}(\ln B + \ln(\ln(\sum_{i\ne i^\star} \frac{1}{\Delta_i^2})) + C') + C''.
\end{aligned}
$$

In order to prove Lemma 11, we first present two lemmas for bounding stopping time for a single armed bandit process i.e., we play only the single arm consecutively until the end of the round. Then, we provide how can we decouple stopping time of multi-armed bandit process of UCB policy.

Lemma 12. *Let $\tau(\pi^i, B)$ be a stopping time for the single armed bandit process π^i which chooses only same drafter i throughout the generation (i.e. $a_t = i$ for all t). Then the stopping time can be bounded as:*

$$
\frac{B(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} - 1 < \mathbb{E}[\tau(\pi^i, B)] \le \frac{(B+1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}}. \tag{28}
$$

Proof. One can see the expected number of generated tokens in each round is $\mu_i^c = \frac{1-\alpha_i^{N_{max}+1}}{1-\alpha_i}$ and the remaining number of tokens in the last round is contained in $\{1, 2, \cdots, N_{max}\}$. Now, suppose eq. 28 holds for all $B < B_0$. Then one can

observe:

$$
\begin{aligned}
&\mathbb{E}[\tau(\pi^i, B_0)] \\
&= \mathbb{E}\left[\sum_{j=0}^{N_{max}} \left(\tau(\pi^i, B_0 - 1 - j) + 1\right) \mathbb{P}[r_i^{BE} = j]\right] \\
&\leq \sum_{j=0}^{N_{max}} \frac{(B_0 - j)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j] + 1 \\
&\leq \sum_{j=0}^{N_{max}} \frac{(B_0 + 1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j] \\
&- \frac{(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{E}[r_i^{BE} = j] + 1 \\
&= \sum_{j=0}^{N_{max}} \frac{(B_0 + 1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j].
\end{aligned}
$$

Since it is trivial to see that eq. 28 holds for $B = 1$, by mathematical induction, one can conclude the proof. The lower bound can be proved by the exactly same manner as in the upper bound.

□

Now, we propose a lemma which provides an upper bound on expected stopping time.

Lemma 13. *For MetaSD-UCB algorithm π with given token target sequence length B, expectation of stopping time $\tau(B)$ can be bounded as follows:*

$$
\mathbb{E}[\tau(B)] \leq \mathbb{E}[\tau(\pi^{i^\star}, B)] + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)], \tag{29}
$$

where, $n_i(\pi, B)$ is number of selecting drafter i by policy π during the generation.

Proof. We first prove the upper bound (eq. 29). For policy π with the target sequence length B, define a corresponding process π^u which is defined by extending the process with the new stopping time, which is:

$$
\begin{aligned}
&\tau^u(\pi^u, B) = min\{\tau > 0 \\
&\mid \sum_{t=1}^{\tau} (N_{acc}(a_t^u, t) + 1) \cdot \mathbb{I}[a_t^u = i^\star] \geq B\}.
\end{aligned} \tag{30}
$$

where, $a_t^u = a_t$ for $t \leq \tau(B)$ and $a_t^u = i^\star$ for $\tau(B) < t \leq \tau^u(\pi^u, B)$. In other words, $\tau^u(\pi^u, B)$ is the time where total number of generated tokens by optimal drafter exceeds B. Then, one can see from the construction of π^u and by observing that τ^u does not depend on thenumber of tokens generated by suboptimal drafters,

$$
\mathbb{E}[n_{i^\star}(\pi, B)] \leq \mathbb{E}[n_{i^\star}(\pi^u, B)] = \mathbb{E}[\tau(\pi^{i^\star}, B)]. \tag{31}
$$

□

Proof of Lemma 11 To prove the upper bound, from Lemma 10 and Lemma 13, it can be shown that

$$
\begin{aligned}
&\mathbb{E}[\tau(B)] \leq \mathbb{E}[\tau(\pi^{i^\star}, B)] + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)] \\
&\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)] \\
&\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \mathbb{E}[\ln \tau(B)] \\
&+ (K-1)(1 + \frac{\pi^2}{3}), \\
&\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln \mathbb{E}[\tau(B)] \\
&+ (K-1)(1 + \frac{\pi^2}{3}),
\end{aligned} \tag{32}
$$

where the second inequality holds from Lemma 12, the third inequality holds by Lemma 10, and the last inequality holds from Jensen's inequality. Now, using $\ln(x) \leq \frac{x}{\epsilon} + \ln(\epsilon) - 1$ and taking $\epsilon = \sum_{i \neq i^\star} \frac{16}{\Delta_i^2}$, one can obtain:

$$
\begin{aligned}
\mathbb{E}[\tau(B)] \leq &\frac{(2B+2) \cdot (1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ 2\ln(\sum_{i \neq i^\star} \frac{16}{\Delta_i^2}) - 2 + \\
&(2K-2)(1 + \frac{\pi^2}{3}).
\end{aligned}
$$

If we again put the above equation into the eq. 32, one can obtain:

$$
\begin{aligned}
\mathbb{E}[\tau(B)] &\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln \frac{(2B+2) \cdot (1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ 2\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2}) + C_1 + C_2 \\
&\leq \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln B \\
&+ \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C' + C'',
\end{aligned}
$$

where $C_1, C_2, C', C'' > 0$ are constants that are independent of B and Δ_i.

□

Proof of Theorem 4 The theorem is proved by observing:

$$
\begin{aligned}
&\mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, B)]) \\
&= \mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^{i^\star}, B)] \\
&\leq \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ \sum_{i \neq i^\star} \frac{8}{\Delta_i^2}(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C') \\
&+ C'' - \mathbb{E}[\tau(\pi^{i^\star}, B)]) \\
&< \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} \\
&+ \sum_{i \neq i^\star} \frac{8}{\Delta_i^2}(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C') \\
&+ C'' - \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} - 1 \\
&< \sum_{i \neq i^\star} \frac{8}{\Delta_i^2}(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C') + C'''.
\end{aligned}
$$

Here, $C', C''' > 0$ are constants independent of B and Δ_i. The first equality comes from Lemma 8, the first inequality is from Lemma 11, and the second inequality holds by putting $i^\star$ to the lower bound of Lemma 12.

□

Note that above analysis holds for every $\beta > 0$ in algorithm 2. However, when the target sequence length B is finite, constant terms in the regret bound becomes important which makes the performance of the algorithm dependent on β. We empirically found the optimal β in our experiments. We provide further discussion on using different β in Section G.8.

G.5 Proof of Theorem 2

Concentration inequality Denote empirical mean of the BD and BE rewards as follows.

$$
\mu_{i,t}^{BD} = \frac{1}{n_i(t)} \sum_{\tau=1}^{t} r_{i,\tau} \cdot \mathbb{I}[a_\tau = i],
$$

$$
\mu_{i,t}^{BE} = \frac{1}{n_i(t) N_{max}} \sum_{\tau=1}^{t} N_{acc}(i,t) \cdot \mathbb{I}[a_\tau = i],
$$

where $n_i(t)$ is number of times drafter i is selected until round t and $\mathbb{I}$ is indicator function.

Then, following inequalities can be derived for $\epsilon > 0$:

$$
\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_i^{BE} \geq \frac{\alpha_i - \alpha_i^{N_{max}+1}}{N_{max}(1-\alpha_i)} + \epsilon\right) \\
&\leq \exp\left(-\frac{n_i(t)\epsilon^2}{2Var[r_i^{BE}] + \epsilon}\right),
\end{aligned} \tag{33}
$$

$$
\mathbb{P}\left(\hat{\mu}_i^{BD} \geq \alpha_i + \epsilon\right) \leq \exp\left(-2(N_{max}) n_i(t) \epsilon^2\right). \tag{34}
$$

eq. 33 comes from combining Bernstein's inequality (Lemma 4) with Lemma 5 and eq. 34 is from combining Hoeffding's inequality (Lemma 3) with Lemma 6.

Bandit algorithm guarantee Using concentration inequalities for both rewards, we provide how the bandit signal defined in eq. 6 directly related to our algorithm Algorithm 2. In the proof of Theorem 4, one can observe that bounding number of suboptimal arm selection (Lemma 10) directly related to the regret under the new regret object defined by stopping time (Definition 1). Leveraging above results, the regret upper bound for MetaSD-UCB algorithm with the BD and BE rewards can be proved.

Proof of Theorem 2 For the BD reward, by putting $\beta = \frac{1}{\sqrt{N_{max}}}$ in the UCB algorithm and apply eq. 34, one can directly observe eq. 25 becomes:

$$
\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i^\star,s} \leq \mu_{i^\star} - \frac{1}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{s}}\right) \\
&\leq \exp(-4\ln t) = t^{-4}, \\
&\mathbb{P}\left(\hat{\mu}_{i.n_i} \geq \mu_i + \frac{1}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{n_i}}\right) \\
&\leq \exp(-4\ln t) = t^{-4}.
\end{aligned} \tag{35}
$$

By choosing $l = \lceil \frac{8\ln\tau(B)}{(N_{max})\Delta(\alpha_i)^2} \rceil$, one can see for $n_i \geq l$:

$$
\frac{2}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{n_i}} \leq \Delta_i.
$$

Rest of the proof is same as in Theorem 4 and we can obtain:

$$
\begin{aligned}
\textsc{Reg}(B) \leq \sum_{i \neq i^\star} \frac{8}{(N_{max})\Delta(\alpha_i)^2} \cdot \\
\left(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C'\right) + C,
\end{aligned}
$$

for some constants $C > 0$ and this concludes the proof of Theorem 2. □

BE reward regret For MetaSD-UCB algorithm with BE reward, we can obtain regret upper bound by the following theorem.

Theorem 5. *Define $\Delta_i^{BE} := \mu_{i^\star}^{BE} - \mu_i^{BE}$ where $\mu_i^{BE} = \mathbb{E}[r_i^{BE}]$. If $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$, we can obtain the following regret upper bound for the MetaSD-UCB algorithm using BE reward:*

$$\mathrm{REG}(\pi^{BE}, B) \leq \sum_{i \neq i^\star} \frac{(32\mathrm{Var}[r_{i^\star}^{BE}] + 16)}{(\Delta_i^{BE})^2} \cdot \left(\ln B + \ln\left(\ln\left(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2}\right)\right) + C' \right) + C, \tag{36}$$

where $C, C' > 0$ are constants independent of B, Δ_i^{BE}.

Proof. From eq. 33, one can similarly modify the original proof of the UCB (Auer, 2002).

Then, putting $\epsilon = \sqrt{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4)\ln t}$ into eq. 33 make eq. 25 becomes:

$$\mathbb{P}\left(\hat{\mu}_{i^\star,s} \leq \mu_{i^\star} - \sqrt{\frac{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4)\ln t}{s}}\right) \leq \exp(-4\ln t) = t^{-4},$$
$$\mathbb{P}\left(\hat{\mu}_{i.n_i} \geq \mu_i + \sqrt{\frac{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4)\ln t}{n_i}}\right) \leq \exp(-4\ln t) = t^{-4}. \tag{37}$$

By choosing $l = \lceil \frac{(32\mathrm{Var}[r_{i^\star}^{BE}]+16)\ln \tau(B)}{(\Delta_i^{BE})^2} \rceil$, one can see for $n_i \geq l$:

$$2 \cdot \sqrt{\frac{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4)\ln t}{n_i}} \leq \Delta_i^{BE}.$$

Rest of the proof is similar as in Theorem 4. □

Regret comparison We restate the Corollary 1 formally as follows:

Corollary 2. *For any $n \in \mathcal{N}$, define functions f_n, g_n, h_n on $(0,1)$ by $f_n(x) = \frac{x - x^{n+1}}{1-x}$, $g_n(x) = f_n'(x) = \sum_{s=1}^n s x^{s-1}$, and $h_n(x) = \sum_{s=1}^n s(x^{s-1} - x^{2n-s})$. If $h_n(\alpha_{i^\star}) \geq \frac{g_n(\alpha_{i^\star})^2}{4n\alpha_{i^\star}}$ and $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$, then the regret of our algorithm π^{BE} with the BE reward feedback is upper bounded by some function $f(B)$, where $f(B) > \frac{8}{(N_{max})(\Delta(\alpha_i))^2} \ln B$.*

Proof. One can observe:

$$\frac{(32\mathrm{Var}[r_i^{BE}] + 16)}{(\Delta_i^{BE})^2} \geq \frac{(32\mathrm{Var}[r_i^{BE}])}{(\Delta_i^{BE})^2} > \frac{16}{\Delta(\alpha_i)^2(N_{max})},$$

where first inequality comes from Theorem 3. Now, putting above result with Theorem 2 and Theorem 5, we get the result. □

Note that the better regret upper bound does not always guarantee the better performance since sometimes it is a proof artifact. Since we take quite loose inequalities during the proof of Theorem 5, we can improve the constant factors for BE reward. Still, even with assuming we can use Lemma 3 inequality in BE reward (which has better guarantee then Bernstein's inequality), the result of Corollary 1 still holds which shows the distinction between two reward designs in terms of regret as in Theorem 3.

G.6 Assumption on model alignment

Here, we formally define the assumption on the acceptance rate which is used throughout our analysis.

Assumption 2. *Denote $\alpha_{i,t}$ as the acceptance rate for t-th token generated by i-th model. Then, for any instance of $x^{1:B}$ generated by the target model, $\alpha_{i,t}$'s are i.i.d. from a distribution ν_i with expectation α_i. In other words, following holds for all drafter $i \in [K]$.*

$$\alpha_{i,t} = 1 - d_{TV}\left(p^t(\cdot|x^{1:t-1}), q_i^t(\cdot|x^{1:t-1})\right) \overset{i.i.d.}{\sim} \nu_i, \quad \mathbb{E}[\alpha_{i,t}] = \alpha_i. \tag{38}$$

Above assumption shows that the acceptance rate for each token only depends on the drafter index i. We empirically verify the validity of the assumption by observing the TV distance between a target model and a drafter is well concentrated (F.4).

Experimental evidence To further validate Assumption 2, we provide empirical observations of the BD rewards in actual speculative decoding scenarios. For the experiment, we follow the same experimental setup of multilingual translational task as in Section 4. We set the target task of Japanese to English translation, where we plot of the statistics when using individual specialized drafters used in Section 4. For each experiment, we generate

80 instances where we calculate the mean and the standard deviation of the BD rewards for each SD round number We take the value over the first 10 rounds considering the fact that few instances take more than 10 rounds of SD.

Figure 6 clearly supports the stationary assumption made in Assumption 2 where mean and the variance of the BD reward behaves similar along the speculative decoding round moves on. Here, $[\mu - s, \mu + s]$ is colorized from the mean μ and the standard deviation s.

Assumption 2 assumes i.i.d. of acceptance rate $\alpha_{i,t}$ in every instance and this might include the case where α_i can vary for every generation. However, this does not affect the analysis of Theorem 2 since our algorithm reset the bandit instance in every new generation.

Temperature sampling Note that we make Assumption 2 for any temperature T which includes greedy decoding scenario with $T = 0$. This implies analysis on the regret upper bound in Section G.4 holds with any temperature $T = 0$. This is empirically supported by our experimental results where MetaSD-UCB (algorithm 2) works for both temperature sampling and greedy decoding scenarios. To further support the above claim. we conduct experiments to obtain BD reward statistics with temperature sampling in Figure 7. For the experiment, we set $T = 1$ and follow the same experimental setup in Figure 6. One can observe the reward is concentrated well while maintaining stationarity along the round numbers. This shows Assumption 2 works with any temperature, further validates our theoretical analysis.

Comparison with (Leviathan et al., 2023; Yin et al., 2024) Leviathan et al. (2024) assume fixed value of α_i when calculating expected number of generated tokens in each round. Assumption 2 is more general than this where in our case, variance of the acceptance rate becomes critical factor to obtain a concentration bound as stated in Lemma 5 and Lemma 6 while this is impossible when assuming fixed acceptance rate. Note that Yin et al. (2024) analyze SD in more general case where they provide the expected number of total rejected tokens as follows:

$$\mathbb{E}[N_{rej}] = \sum_{t=1}^{T} \mathbb{E}_{x_{1:t-1} \sim p^t}[d_{TV}(p^t(\cdot|x^{1:t-1}), q_i^t(\cdot|x^{1:t-1})] \tag{39}$$

This is general than Assumption 2 where we assume previous context $x^{1:t-1}$ does not affect the TV distance between target model and a drafter. Relaxing the assumption and considering context-dependent reward distribution will be related to a contextual bandit problem (Li et al., 2010) while we leave this as a future work.

G.7 Randomness of the target sequence length B

We consider general scenario where we take all possible instances generated by a target model when using temperature sampling with $T > 0$. In this scenario, we can define the expected regret over the probability space induced by the target model. To do so, we first provide a formal definition of a target sequence length B.

Definition 4 (Target sequence length B)**.** *Target sequence length B is a stopping time which is defined as follows:*

$$B = \min\{t \in \mathbb{N} : x^t = EOS\}, \tag{40}$$

where $x^t \sim p^t(\cdot|x^{1:t-1})$ with p^t being a probability distribution from the target model given context $x^{1:t-1}$ and EOS refers to the end of sentence token.

According to Definition 4, target sequence length is a random variable (a stopping time). With this, one can observe the following lemma holds:

Lemma 14. *For $b \in \mathbb{N}$,*

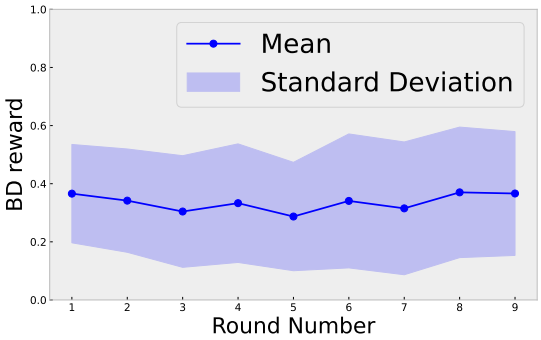
$$\mathbb{P}(B = b) = \mathbb{E}_{x^{1:b-1} \sim p} \prod_{t=1}^{b-1} (1 - p^t(EOS|x^{1:t-1}) \cdot p^b(EOS|x^{1:b-1}) \tag{41}$$

Where, $p^t(\cdot|x^{1:t-1})$ refers to the conditional probability distribution from a target model for t-th token generation when given context $x^{1:t-1}$. Moreover, expectation of a target sequence length becomes:

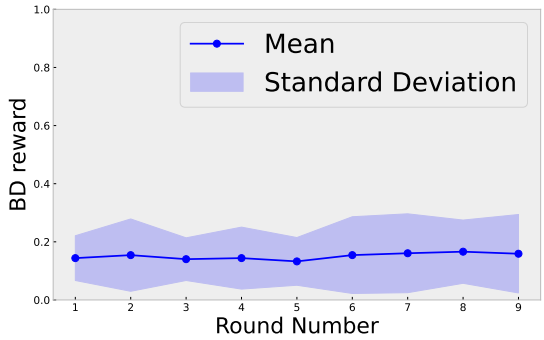
$$\mathbb{E}[B] = \mathbb{E}_p(B) = \sum_{b=1}^{\infty} b \cdot \mathbb{P}(B = b). \tag{42}$$

Here, $\mathbb{E}_p$ denotes the expectation taken over the probability distribution induced by the target model p.

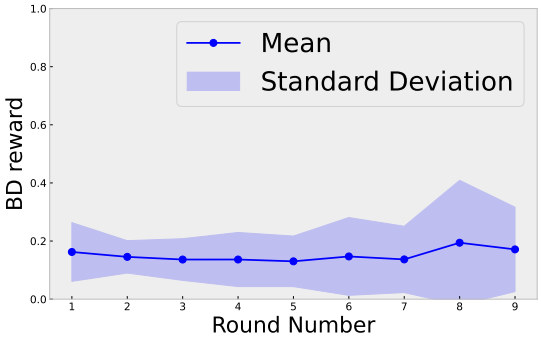
Then the expected stopping time which includes every instance of token generation realization given a context can be analyzed with the following objective.



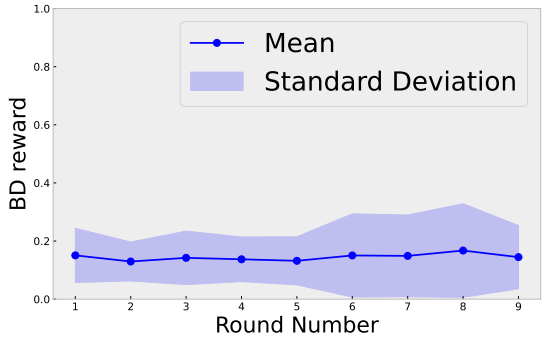
(a) Drafter 1



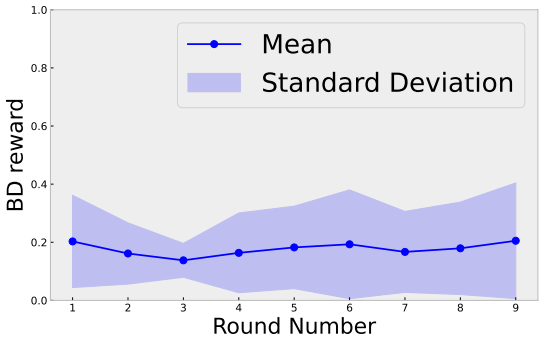
(b) Drafter 2



(c) Drafter 3



(d) Drafter 4



(e) Drafter 5

Figure 6: Empirical measurement of BD reward statistics along speculation rounds in greedy decoding ($T = 0$) scenario.

Definition 5 (Expected stopping time regret)**.**

$$\textsc{Reg}(\pi, B) = \mathbb{E}_{p,\pi}\left[\tau(\pi, B)\right] - \mathbb{E}_{p,\pi^\star}\left[\tau(\pi^\star, B)\right], \tag{43}$$

where, $\mathbb{E}_p$ denotes the expectation taken over from a probability space induced by the randomness of target model generation and $\mathbb{E}_\pi, \mathbb{E}_{\pi^\star}$ refers to the expectation taken over from the probability space generated by a bandit policy π and the optimal policy $\pi^\star$ respectively.

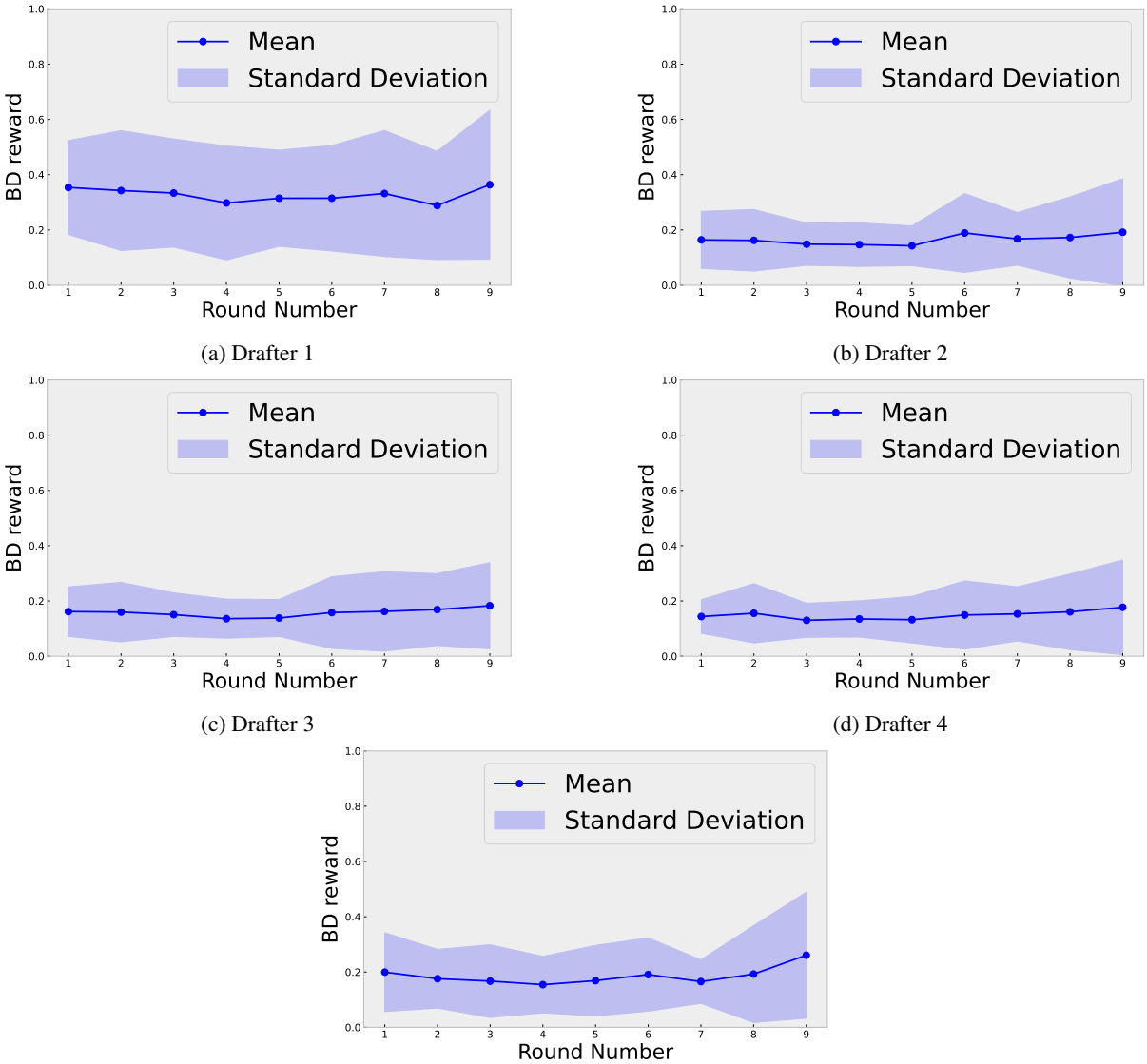
In order to analyze the expectation of the stopping time regret which includes the randomness of B, we need a more stronger assumption than in Assumption 1 which is stated as follows.

Assumption 3. *Denote $\alpha_{i,t}$ as the acceptance rate for t-th token generated by i-th model. Then, for any instance $x^{1:B}$ generated by the target model, $\alpha_{i,t}$'s are i.i.d. from a distribution ν_i with expectation α_i. In other words, following holds for all drafter $i \in [K]$.*

$$\alpha_{i,t} = 1 - d_{TV}\left(p^t(\cdot|x^{1:t-1}), q_i^t(\cdot|x^{1:t-1})\right) \overset{i.i.d.}{\sim} \nu_i, \quad \mathbb{E}[\alpha_{i,t}] = \alpha_i. \tag{44}$$

Moreover, α_i is independent of B and its conditional expectation over the events with given B is same for every B.

The above assumption implies acceptance rate for each drafter is i.i.d. from a stationary distribution of a given instance and its mean value is independent of B. Now, with the generalized re-



(a) Drafter 1

(b) Drafter 2

(c) Drafter 3

(d) Drafter 4

(e) Drafter 5

Figure 7: Empirical measurement of BD reward statistics along speculation rounds in temperature sampling $T = 1$.

gret objective and Assumption 3, one can obtain regret upper bound in terms of expectation of total generated tokens.

Theorem 6 (Expected stopping time regret)**.** *Under Assumption 3, following regret bound holds for Meta-UCB with general stopping time regret:*

$$\textsc{Reg}(\pi, B) < \sum_{i \neq i^\star} \frac{8}{(N_{max})\Delta(\alpha_i)^2} \cdot \left(\ln\left(\mathbb{E}[B]\right) + \ln\left(\ln\left(\sum_{i \neq i^\star} \frac{1}{\Delta(\alpha_i)^2}\right)\right) + C' \right) + C. \tag{45}$$

Here, $C, C' > 0$ are again constants that are independent from B and $\Delta(\alpha_i)$.

Proof. Since drafter selection from the policy π is independent from B under Assumption 3, we can decouple eq. 43 as follows:

$$\textsc{Reg}(\pi, B) = \mathbb{E}_B[\mathbb{E}_\pi\left[\tau(\pi, B)\right] - \mathbb{E}_{\pi^\star}[\tau(\pi^\star, B)]], \tag{46}$$

where first expectation is taken over with respect to a probability distribution of B generated from p. Using Jensen's inequality and combining with Theorem 2, we get the result. □

G.8 Further analysis on hyper-parameter β

Although original UCB-1 algorithm in (Auer, 2002) is based on using fixed value of $\beta = 1$, following works (Audibert et al., 2009; Bubeck, 2010) show the regret can indeed be dependent on the exploration parameter β. We provide a general results which includes a hyperparameter β in

MetaSD-UCB algorithm. In the following, we borrow the analysis of (Bubeck, 2010) for the general version of Theorem 2 that includes β.

Theorem 7 (Regret upper bound containing β)**.** *For $\beta > 0.5$ and with Assumption 2, the regret upper bound in Theorem 2 can be generalized as follows:*

$$\textsc{Reg}(\pi, B) < \sum_{i \neq i^\star} \frac{8\beta^2}{(N_{max})\Delta(\alpha_i)^2} \cdot \left(\ln B + \ln(\ln(\sum_{i \neq i^\star} \frac{1}{\Delta_i^2})) + C' \right) + C. \tag{47}$$

Proof. The proof is based on modifying Lemma 10 to the equation (2.15) in (Bubeck, 2010) which is stated here for the completeness.

$$\mathbb{E}[n_i(\tau(B))|\tau(B)] \leq \frac{8\beta^2 \ln \tau(B)}{\Delta_i^2} + 1 + \frac{4}{\ln(2\beta^2 + \frac{1}{2})} \left(\frac{2\beta^2 + \frac{1}{2}}{2\beta^2 - \frac{1}{2}} \right)^2 \tag{48}$$

Rest of the procedure is same with Theorem 2. □

Note that extra β^2 appears in the regret bound and supports and constant term can arbitrarily blow up when β becomes close to the $\frac{1}{2}$ by the right term in eq. 48. We refer (Bubeck, 2010) for further details of the calculations.

H Extended scenarios for the MetaSD framework

Our MetaSD framework is universal as it can incorporate various bandit algorithms tailored for different scenarios. However, establishing optimality guarantees for existing algorithms in this framework requires careful analysis or one should look for the different algorithm designs. This is due to two key distinctions in our problem formulation: (i) stochastic stopping time, and (ii) a new regret objective defined in terms of this stopping time (Definition 1).

This section explores two distinct scenarios and introduces possible algorithms for each. First, we address a scenario when switching costs is not negligible anymore. In MetaSD framework, this happens when substantial computational or memory overhead is incurred when changing drafters. Second, we consider non-stationary environment where the characteristics of the context change within a one generation. Finally, we briefly discuss on other possible extensions of our framework.

H.1 Switching costs

Switching costs for multiple drafters In order to use multiple drafters in SD, one need to replace all missing key-value(KV) cache values for the model whenever switching one drafter to another. Reading and writing KV cache is one of the factor which can decrease the inference speed, and we define any decrease of inference speed by changing drafter as the switching cost. Formally, switching cost is defined as $\lambda(a_t, t) = \lambda\left(l(t) - l(\tau_i(t))\right) \cdot \mathbb{I}[a_{t-1} \neq a_t]$ where $l(t)$ is number of processed tokens by the target model in round t, $\tau_i(t)$ is the latest round where i-th drafter is selected before round t, $\mathbb{I}$ is an indicator function, and λ is a constant. we first define the pseudo regret objective in the presence of switching costs.

Definition 6. *With bandit policy π and the given budget B, we define the regret as follows:*

$$\textsc{Reg}_{switch}(\pi, B, \lambda) = \mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, b)] + \sum_{t=2}^{\tau(B)} \lambda_t \mathbb{P}(a_{t-1} \neq a_t). \tag{49}$$

To minimize the above regret, observe $\lambda(\pi, B) = \lambda \sum_{t=1}^{\tau(B)} \lambda(a_t, t) = \lambda \sum_{i=1}^{K} B_i$, where B_i's are total number of tokens generated by the i-th drafter after the final round. Intuitively, this implies that total cost decreases when employing elimination-type of algorithms (Audibert and Bubeck, 2010; Karnin et al., 2013), which successively eliminate sub-optimal drafters and exclude those drafters from future selection. Consequently, the total regret $\textsc{Reg}_{switch}(B, \lambda)$ can be reduced from early elimination of poor-performed drafters. However, regret can still increase if the best drafter is mistakenly eliminated early on. Therefore, it is essential to strike a balance between elimination-based algorithms and standard MAB algorithms. For this, we design a new algorithm **Pure Exploration-Then-Commit** (PETC) in Algorithm 4 which effectively balances these two approaches.

PETC (Algorithm 4) divides the MetaSD into two phases. In the first phase $l < B_0$, the algorithm tries to eliminate sub-optimal drafters as quickly as possible. In the bandit literature, this is related to the pure exploration (or best arm identification) problem (Lattimore and Szepesvári, 2020) and we select using SH Algorithm 5 for our analysis. After the exploration period for estimating the

Algorithm 4: Pure exploration-then-commit (PETC)

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration rounds B_0.

for $l = 1, 2, ..., B_0$ **do**
  Run SH algorithm with budget B_0 (in Algorithm 5)
end for
$\hat{i}_\star$ be the survived index.
while $l < B$ **do**
  SD with a single drafter $\hat{i}_\star$.
end while

Algorithm 5: Sequential Halving (SH) (Karnin et al., 2013)

INPUT Total budget T, drafter pool $[K]$

Initialize $S_0 \leftarrow [K]$
for $t = 0, 1, \ldots, \lfloor \log_2(K) \rfloor - 1$ **do**
  Pull each drafter in S_t for $n_t = \left\lfloor \frac{T}{|S_t| \lceil \log_2(K) \rceil} \right\rfloor$ additional times
  $R_t(i) \leftarrow \sum_{j=1}^{n_t} r_{i,j}$ for $i \in S_t$
  Let σ_t be a bijection on S_k such that $R_t(\sigma_t(1)) \leq R_t(\sigma_t(2)) \leq \ldots \leq R_t(\sigma_t(|S_t|))$
  $S_{k+1} \leftarrow [i \in S_k | R_t(\sigma_t(i)) \leq R_t(\sigma_t(\lceil |S_k|/2 \rceil))]$
end for

OUTPUT Singleton element of $S_{\lfloor \log_2(K) \rfloor}$

best drafter, the algorithm exclusively selects this drafter for the remaining rounds.

Now, we provide how to find the optimal B_0 which by the following theorem:

Theorem 8 (Regret upper bound on PETC)**.** *By choosing $B_0 = c \cdot \ln B$ for some constant $c > 0$ and using Algorithm 5 for the pure exploration in the for the first phase in Algorithm 4, $\text{REG}_{switch}(\pi, B, \lambda) \leq O(\ln B)$ holds.*

Proof. First, we can decompose the regret as:

$$\text{REG}_{switch}(\pi, B, \lambda) = \sum_{t=1}^{\tau(B_0)} \text{REG}(\pi, t) + \sum_{t=\tau(B_0)+1}^{\tau(B)} \text{REG}(\pi, t) + S_T,$$

where $\text{REG}(\pi, t)$ denotes original regret objective eq. 1 for one round t and S_T denotes the total switching cost. First term can be bounded by the stopping time of selecting the worst drafter every round until B_0 which can be bounded by $\tau(B_0) = O(\ln B)$ according to Lemma 12. To bound the second term, we borrow Theorem 4.1 in (Karnin et al., 2013), where they prove the probability of Sequential Halving algorithm to select the suboptimal arm after B_0 round can be bounded by $3 \log_2 K \cdot \exp(-\frac{B_0}{8 H_2 \log_2 K})$, where $H_2 := \max_i \frac{i}{\Delta_i^2}$. Then we have

$$\sum_{t=\tau(B_0)+1}^{\tau(B)} \text{REG}(\pi, t) \leq \tau(\pi^{i_w}, B) \cdot 3 \log_2 K \cdot \exp(-\frac{B_0}{8 H_2 \log_2 K}) = O(\ln B),$$

where i_w denotes the worst drafter, $\tau(\pi^{i_w}, B)$ denotes the stopping time for generating B tokens using only the worst drafter. The last term is bounded by $\lambda B_0 = O(\ln B)$ and this concludes the proof. □

Here, we can improve constant term in regret upper bound in Theorem 8 by controlling c according to the switching cost λ and given budget B or we may use more advanced proof techniques in the best arm identification literature such as in (Zhao et al., 2023). We leave these as a future work.

H.2 Non-stationary environment

In real-world scenarios, the reward distribution for each drafter may evolve over time and past information becomes less relevant for decision-making. This phenomenon, referred to as non-stationarity, challenges traditional MAB algorithms that operate under the assumption of stationary reward distributions. In SD, non-stationarity can stem from various factors. For example, during a long-form text generation task, the optimal drafter may change as the topic or style of the text evolves. Consider the prompt: 'Please summarize and reason about the following article on climate change...'. Initially, a drafter specialized in summarization might be most effective. However, as the generation progresses towards the reasoning part, a drafter trained on logical reasoning tasks could become more suitable.

Non-stationary MetaSD Standard analyses of non-stationary bandits (Auer et al., 2002; Kocsis and Szepesvári, 2006; Garivier and Kaufmann, 2016) often define L to quantify the number of times the reward distributions change over T

rounds. Another line of work (Slivkins and Upfal, 2008; Besbes et al., 2014) quantifies the non-stationarity using V, the total variation of the means. In both cases, the regret (which is often called as dynamic regret) is defined as the cumulative expected difference between the rewards of the optimal arm and the selected arm at each round.

$$\text{REG}(\pi, B, L) = \sum_{t=1}^{\tau(B)} (\max_{i \in [K]} \mu_{i,t} - \mathbb{E}[\mu_{a_t,t}]) \quad (50)$$

where, as before, B is the number of total tokens we have to generate, $\mu_{i,t}$ is the mean reward of choosing drafter i in t-th round, and $\tau(B)$ is the total round. However, the regret upper bound on eq. 50 does not always guarantee the performance of the SD as we discussed in Section 3.1. Instead, we can use our original regret objective using stopping time in Definition 1 without any modification.

Here, we introduce two types of algorithms within our MetaSD framework: Discounted-UCB (D-UCB) algorithm (Kocsis and Szepesvári, 2006) (Algorithm 6) and Sliding-window UCB (Garivier and Moulines, 2011) (Algorithm 8). Discounted UCB-SD estimates mean reward by computing the mean of discounted cumulative rewards as shown in the line 9 of Algorithm 6. By assigning less weight to the past observations, the algorithm finds a balance between accumulating knowledge and adapting to the changing environment. Similarly, sliding-window UCB utilizes a fixed-length window to calculate mean reward as demonstrated in the line 9-10 of Algorithm 8. By focusing only on recent information, it is also expected to achieve a balance with careful choose of the window size τ (Garivier and Moulines, 2011).

One interesting point is that in the non-stationary MetaSD problem, the definition of non-stationarity L does not fit naturally into our problem. The reason behind this is that under non-stationary context generations, number of distribution changes happen at the token level, not the round level. This can disrupt existing regret analysis because a single round might involve multiple reward distribution changes (e.g., one round of speculative decoding could have two changing points). Whether above algorithms maintain optimal regret bounds in our regret definition in this non-stationary setting presents an interesting direction for future theoretical analysis.

Algorithm 6: Discounted UCB in MetaSD

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration strength β, decaying parameter γ.

1: $t \leftarrow 0$
/* Phase 1: Meta-draft each drafter in $[K]$ once and do one round of speculative decoding. */
2: **for** $i \in [K]$ **do**
3: Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 4)
4: $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t + 1$
5: **end for**
/* Phase 2: Meta-draft the draft following the UCB bandit until target sequence length B */
6: **while** $l < B$ **do**
7: $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta \sqrt{\frac{2 \ln t}{n_i}}$
8: Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 4)
9: $\hat{\mu}_{a_t,t} = \frac{1}{n_{a_t}} \sum_{s=1}^{t} \gamma^{t-s} r_{a_s,s} \mathbb{I}[a_s = a_t]$
10: $n_{a_t}, l, t \leftarrow n_{a_t} + 1, l + N_{acc}(a_t,t) + 1, t + 1$
11: **end while**

Algorithm 7: MetaSD-EXP3 (Auer et al., 2002)

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, $\gamma \in (0, 1]$

1: $t \leftarrow 0$, $w_t(i) \leftarrow 1$ for $i = 1, \ldots, K$
2: **while** $l < B$ **do**
3: $p_t(i) = (1-\gamma) \frac{w_t(i)}{\sum_{i=1}^{K} w_t(i)} + \frac{\gamma}{K}$ for $i = 1, \ldots, K$.
4: Draw a_t randomly according to the probabilities $p_t(1), \ldots, p_t(K)$.
5: Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 4)
6: **for** $j = 1, \ldots, K$ **do**
7:

$$\hat{r}_{j,t} = \begin{cases} r_{j,t}/p_t(j) & \text{if } j = a_t \\ 0 & \text{otherwise,} \end{cases}$$

$$w_{t+1}(j) = w_t(j) \exp\left(\frac{\gamma \cdot \hat{r}_{j,t}}{K}\right)$$

8: **end for**
9: $l, t \leftarrow l + N_{acc}(a_t,t) + 1, t + 1$
10: **end while**

Experiments To further demonstrate the effectiveness of the MAB approach, we conduct experiments on a non-stationary translation task, where each query required translating two different lan-

guages (French and Chinese) into English. This presents a challenging task, as even well-trained router-based algorithms struggle to capture the shifting context during generation. The result is provided in Table 19.

The result clearly demonstrate the effectiveness of the MAB approach, where MetaSD-UCB with $\alpha = 0.1$ achieves a speedup ratio of 1.722, which even exceeds the performance of the optimal drafter with a speedup ratio 1.581. This improvement arises because MAB dynamically adapts to the best drafter in the current environment through a combination of exploration and exploitation.

For example, in a multilingual scenario, where the task is "Translate French and German to English, respectively," classification-based routing would be limited by the performance of the best single specialization (i.e., the speedup ratio of the optimal drafter as its upper bound). In contrast, MetaSD-UCB surpasses this upper bound by effectively adapting its policy dynamically, demonstrating a unique advantage over classification-based methods in non-stationary environments.

The experiments are performed using a single RTX 3090 GPU, and we follow the same experimental setup of MetaSpS-UCB in Section 4. In Table 19, Upper Bound refers to the performance achieved by an ideal router-based approach with oracle-provided labels.

H.3 Other possible scenarios

Adversarial environment EXP3 (Auer et al., 2002) is designed to handle adversarial changes of reward distributions by continuously updating its estimates of the arm rewards and adjusting its exploration strategy accordingly. It achieves this by maintaining a probability distribution over the arms and exponentially weighting the rewards based on their recent performance. By incorporating EXP3 into our framework (Algorithm 7), we can enable the system to adapt to evolving reward distributions and dynamically select the optimal drafter even in adversarial environments. We utilize this algorithm as a baseline in our experiments.

I Further discussion

I.1 Is scaling up drafter size always better?

While increasing the drafter size might seem like a straightforward path to improved performance, it can be less efficient than our MetaSD approach, especially considering memory bandwidth constraints. Larger models demand more memory for storing weights and activations, increasing data movement between memory and processing units. This can become a bottleneck, particularly in high-performance computing where memory bandwidth is often a limiting factor. It is also discussed in (Yi et al., 2024) in SD scenarios. Moreover, this phenomenon is well-illustrated by the roofline model, which highlights the trade-off between computational intensity and memory bandwidth (Cai et al., 2024). As model size increases, computational intensity might improve, but the memory bandwidth demands can quickly limit overall speedup.

In contrast, MetaSD utilizes multiple smaller drafters with lower individual memory requirements. By efficiently switching between these drafters, MetaSD can achieve comparable or superior performance to a single large drafter while mitigating the memory bandwidth bottleneck. This is because, despite having multiple drafters, MetaSD only utilizes one drafter for computation at any given time. Thus, the memory bandwidth requirement does not scale with the combined size of all drafters, but rather with the size of the individual drafter being used. Provided sufficient GPU DRAM, this approach does not have any bottleneck compared to the single drafter SD. Furthermore, MetaSD offers the flexibility to incorporate diverse drafters with specialized capabilities. This specialization can be more effective than simply increasing the size of a single general-purpose drafter, particularly for tasks demanding domain-specific knowledge.

I.2 Out-of-domain (OOD)

We evaluate MetaSD on Alpaca-Finance (Bhartia, 2023) and RAG (Xia et al., 2024b), which fall outside the training domains of specialized drafters. As shown in Table 11 (in Section F.8), MetaSD outperforms OFA and most specialized drafters, demonstrating its ability to generalize without relying on predefined domain similarities. Unlike similarity-based selection, which incurs high inference costs for long inputs and is prone to misclassification, MetaSD dynamically adapts at the token level, ensuring efficient and robust routing. In heterogeneous drafter settings, where training descriptions are incomplete or unavailable, MetaSD remains effective, highlighting its scalability and adaptability in real-world scenarios.

Table 19: Results of the non-stationary translation task.

	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	Upper Bound	MetaSpS-UCB
Block Efficiency	1.668	1.722	1.485	1.759	1.803	1.803	1.951
Speedup Ratio	1.429	1.492	1.289	1.539	1.581	1.581	1.722

Algorithm 8: Sliding-window UCB in MetaSD

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration parameter β, window size τ.

$t \leftarrow 0$

/* Phase 1: Meta-draft each drafter in $[K]$ once and do one round of speculative decoding. */

for $i \in [K]$ **do**

  Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 4)

  $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t + 1$

end for

/* Phase 2: Meta-draft the draft following the UCB bandit until target sequence length B */

while $l < B$ **do**

  $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta \sqrt{\frac{2 \ln t}{n_i}}$

  Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 4)

  $\hat{\mu}_{i,t} \leftarrow \frac{1}{n_i(t)} \sum_{s=t-\tau+1}^{t} r_{a_s,s} \mathbb{I}[a_s = i] \; \forall i \in [K]$

  $n_i(t) \leftarrow \sum_{s=t-\tau+1}^{t} \mathbb{I}[a_s = i] \; \forall i \in [K]$

  $l, t \leftarrow l + N_{acc}(a_t,t) + 1, t + 1$

end while

I.3 Computational overhead analysis

Training overhead While specialization may require additional training efforts compared to an OFA (One-size-Fits-All) drafter, we emphasize that our approach is designed to handle real-world scenarios where heterogeneous drafters already exist in public repositories. MetaSD focuses on optimizing the utilization of such heterogeneous drafters, dynamically selecting the most suitable drafter during inference. This shifts the problem from retraining models to developing an effective strategy for utilizing pre-existing resources. Therefore, while training specialized drafters may involve additional costs in certain cases, the broader applicability and versatility of MetaSD provide substantial practical value. Additionally, the cost of training drafters is a general challenge shared across the speculative decoding research domain, not limited to our work.

Inference memory-bandwidth efficiency The inference memory-bandwidth efficiency of MetaSD remains comparable to single-drafter methods. Although MetaSD employs multiple drafters, the additional memory requirements are minimal. Specifically, MetaSD increases DRAM usage by only 2 GB (from 17 GB to 19 GB), as the drafters' weights are preloaded into DRAM. However, this does not affect VRAM bandwidth, as only the active drafter interacts with VRAM during inference. As a result, the VRAM bandwidth demands remain identical to those of single-drafter methods. This efficient memory management ensures that MetaSD maintains competitive performance without introducing significant overhead.

By ensuring that only the active drafter interacts with the VRAM, MetaSD maintains parity with single-drafter approaches in terms of VRAM bandwidth demands.

Serving complexity Using multiple drafters in MetaSD does not inherently increase serving complexity. Modern distributed systems already employ model parallelism techniques to allocate workloads across multiple GPUs effectively. In MetaSD, drafters are evenly distributed across GPUs, with each GPU independently handling its assigned drafter without added coordination costs. This design ensures the following:

- Load balancing: Drafters are distributed across GPUs based on their assigned tasks, maintaining equivalent complexity to single-

drafter systems.

- Minimal communication overhead: MetaSD requires no additional inter-GPU communication beyond standard model parallelism setups.

Justification of overhead The modest increase in DRAM memory usage (+2 GB) and marginal training cost for specialized drafters is justified by the significant performance gains achieved through adaptive optimization. MetaSD dynamically selects the most suitable drafter for each task, consistently outperforming single-drafter methods across diverse scenarios, as highlighted in our experimental results. Furthermore, MetaSD addresses an important real-world challenge: effectively utilizing publicly available, pre-trained heterogeneous drafters. By providing a generalizable strategy for optimizing these resources, MetaSD adds practical value beyond specialized retraining, supporting diverse and evolving task requirements.

I.4 Regret upper bound for MetaSD-UCB

Theorem 2 provides a regret upper bound for MetaSD-UCB, demonstrating that the number of rounds required to identify the optimal drafter is inversely proportional to the predefined draft length N_{max}. This aligns with the intuition that longer drafts provide more information about the relative performance of each drafter, leading to faster convergence towards the optimal choice. The logarithmic dependence on the target sequence length B further highlights the efficiency of MetaSD-UCB in minimizing regret. These theoretical guarantees are supported by our empirical observations, where MetaSD-UCB consistently demonstrates strong performance and rapid convergence towards the best-performing drafter.