Recursive Speculative Decoding: Accelerating LLM Inference via Sampling Without Replacement

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee*, **Christopher Lott***
Qualcomm AI Research†

Abstract

Speculative decoding is an inference-acceleration method for large language models (LLMs) where a small language model generates a draft-token sequence which is further verified by the target LLM in parallel. Recent works have advanced this method by establishing a draft-token tree, achieving superior performance over a single-sequence speculative decoding. However, those works independently generate tokens at each level of the tree, not leveraging the tree's entire diversifiability. Besides, their empirical superiority has been shown for fixed length of sequences, implicitly granting more computational resource to LLM for the tree-based methods. None of the existing works has conducted empirical studies with fixed target computational budgets despite its importance to resource-bounded devices. We present Recursive Speculative Decoding (RSD), a novel tree-based method that samples draft tokens *without* replacement and maximizes the diversity of the tree. During RSD's drafting, the tree is built by either *Gumbel-Top-k trick* that draws tokens without replacement in parallel or *Stochastic Beam Search* that samples *sequences* without replacement while early-truncating unlikely draft sequences and reducing the computational cost of LLM. We empirically evaluate RSD with Llama 2 and OPT models, showing that RSD outperforms the baseline methods, consistently for fixed draft sequence length and in most cases for fixed computational budgets at LLM.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Zhang et al., 2022; Brown et al., 2020; Achiam et al., 2023; Jiang et al., 2023) have gained popularity due to their outstanding achievements with high-quality text generation, which has drastically increased demands for faster text generation. However, auto-regressive nature of LLMs limits text generation to produce a single token at a time and often suffers from memory-bandwidth bottleneck, which leads to slower inference (Shazeer, 2019).

Speculative decoding (Chen et al., 2023; Leviathan et al., 2023) has emerged as a solution for LLM inference acceleration by leveraging the innate parallelizability of the transformer network (Vaswani et al., 2017). This decoding method utilizes a draft model, i.e., a smaller language model, to auto-regressively generate a sequence of draft tokens with a significantly lower cost and latency, followed by the target LLM producing the token-wise probability distributions in parallel. Rejection sampling then verifies those draft tokens, recovering the sequence distribution by auto-regressive decoding with the target model. As speculative decoding uses a single sequence of draft tokens, one needs to increase the draft-sequence length to better exploit LLM's parallelizability. However, the longer draft sequence may slow down the overall inference in practice due to the computational overhead caused by additional auto-regressive decoding steps from the draft model, possibly decelerating the target model process due to the increased number of draft tokens.

*Correspondence to Wonseok Jeon (wjeon@qti.qualcomm.com), Mingu Lee (mingul@qti.qualcomm.com), Christopher Lott (clott@qti.qualcomm.com)

†Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Recent works on tree-based speculative decoding (Sun et al., 2023; Miao et al., 2023) have achieved better diversity and higher acceptance rate via multiple draft-token sequences. Despite promising results, their decoding methods independently sample the draft tokens, often harming the diversity of the tree when samples overlap. Also, their experiments have been conducted for the fixed length of draft-token sequences across decoding methods, implicitly requiring more computational resource to the target model when using tree-based methods. To the best of our knowledge, no prior work has thoroughly investigated the performance of single-sequence and tree-based speculative decoding methods with fixed target computational budget, which has practical importance for resource-bounded devices.

We propose **R**ecursive **S**peculative **D**ecoding (RSD), a novel tree-based speculative decoding algorithm that fully exploits the diversity of the draft-token tree by using sampling without replacement. We summarize our contributions as below:
Theoretical contribution. We propose *recursive rejection sampling* capable of recovering the target model's distribution with the sampling-without-replacement distribution defined by the draft model.
Algorithmic contribution. We present RSD which builds draft-token tree composed of the tokens *sampled without replacement*. Two tree construction methods, **RSD** with **C**onstant branching factors (RSD-C) and **RSD** with **S**tochastic Beam Search (RSD-S) (Kool et al., 2019), are proposed.
Empirical contribution. Two perspectives are considered in our experiments: ***(Exp1)*** *performance for fixed length of draft sequence*, which is also widely considered in previous works (Sun et al., 2023; Miao et al., 2023), and ***(Exp2)*** *performance for fixed target computational budget*, where we compared methods with given size of the draft-token tree. RSD is shown to outperform the baselines consistently in ***(Exp1)*** and for the majority of experiments in ***(Exp2)***.

2 Background

Let us consider a sequence generation problem with a set $\mathcal{X}$ of tokens. We also assume that there is a target model characterized by its conditional probability $q(x_{i+1}|x_{1:i}) := \Pr\{X_{i+1} = x_{i+1}|X_{1:i} = x_{1:i}\}$, $i \in \mathbb{N}$ for $x_{1:i} := (x_1, ..., x_i)$, where $X_1, ..., X_{i+1} \in \mathcal{X}$ and $x_1, ..., x_{i+1} \in \mathcal{X}$ are random tokens and their realizations, respectively. Given an input sequence $X_{1:t} = x_{1:t}$, we can auto-regressively and randomly sample an output sequence $X_{t+1:t+i}$ for $i \in \mathbb{N}$, i.e., $X_{t+i+1} \sim q(\cdot|X_{1:t+i})$.

Speculative decoding. Auto-regressive sampling with modern neural network accelerators (e.g., GPU/TPU) is known to suffer from the memory-bandwidth bottleneck (Shazeer, 2019), which prevents us from utilizing the entire computing power of those accelerators. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) addresses such issue by using the target model's parallelizability. It introduces a (small) draft model which outputs $p(\hat{X}_{i+1}|\hat{X}_{1:i}) := \Pr\{\hat{X}_{i+1} = \hat{x}_{i+1}|\hat{X}_{1:i} = \hat{x}_{1:i}\}, i \in \mathbb{N}$. Speculative decoding accelerates the inference speed by iteratively conducting the following steps:
1) Draft token generation: For an input sequence $X_{1:m} = x_{1:m}$ and the draft sequence length L, sample draft tokens $\hat{X}_{n+1} \sim p(\cdot|X_{1:m}, \hat{X}_{m+1:n})$ auto-regressively for $n = m, ..., m+L-1$ (where $\hat{X}_{m+1:m} = \emptyset$).
2) Evaluation with target model: Use the target model to compute $q(\cdot|X_{1:m}, \hat{X}_{m+1:n}), n = m, ..., m+L$ in parallel.
3) Verification via rejection sampling: Starting from $n = m+1$ to $m+L$, sequentially accept the draft token $\hat{X}_n$ (i.e., $X_n = \hat{X}_n$) with the probability $\min\{1, \frac{q(\hat{X}_n|X_{1:n-1})}{p(\hat{X}_n|X_{1:n-1})}\}$. If one of the draft tokens $\hat{X}_n$ is rejected, we sample $X_n \sim q_{\text{res}}(\cdot|X_{1:n-1})$, where the residual distribution is defined by $q_{\text{res}}(\cdot|\tau) := \text{Norm}[[q(\cdot|\tau) - p(\cdot|\tau)]^+]$, for $[f]^+ := \max\{0, f(\cdot)\}$ and $\text{Norm}[f] := \frac{f}{\sum_{x' \in \mathcal{X}} f(x')}$.
If all draft tokens are accepted ($X_n = \hat{X}_n$ for $n = m+1, ..., m+L$), sample an extra token $X_{m+L+1} \sim q(\cdot|X_{1:m+L})$.
Chen et al. (2023) and Leviathan et al. (2023) have shown that the target distribution can be recovered when rejection sampling is applied.

Tree-based speculative decoding. One can further improve the sequence generation speed by using multiple draft-token sequences, or equivalently, a tree of draft tokens. *SpecTr* (Sun et al., 2023) is a tree-based speculative decoding algorithm motivated by the Optimal Transport (OT) (Villani et al., 2009). It generalizes speculative decoding with K i.i.d. draft tokens $\hat{X}^{(k)} \sim p, k = 1, ..., K$, while recovering the target distribution q. To this end, a K-sequential draft selection algorithm (K-SEQ) was proposed, where the algorithm decides whether to accept K draft tokens $\hat{X}^{(k)}, k = 1, ..., K$, or not with the probability $\min\{1, \frac{q(\hat{X}^{(k)})}{\gamma \cdot p(\hat{X}^{(k)})}\}, \gamma \in [1, K]$. If all draft tokens are rejected, we use a token drawn from the residual distribution

$$\mathrm{Norm}\left[q - \min\left\{p, \frac{q}{\gamma}\right\} \frac{1 - (1 - \beta_{p,q}(\gamma))^K}{\beta_{p,q}(\gamma)}\right]$$

for $\beta_{p,q}(\gamma) := \sum_{x \in \mathcal{X}} \min\{p(x), q(x)/\gamma\}$. *SpecInfer* also used the draft-token tree to speed up the inference *with multiple draft models* $p^{(k)}, k = 1, ..., K$ (Miao et al., 2023). During the inference of SpecInfer, all draft models generate their own draft tokwns independently and create a draft-token tree collectively through repetetion. For draft verification, *multi-round rejection sampling* is used to recover the target distribution, where we determine whether to accept one of the draft tokens or not with probability $\min\{1, \frac{q^{(k)}(\hat{X}^{(k)})}{p^{(k)}(\hat{X}^{(k)})}\}$ with distributions $q^{(1)} := q$ and $q^{(k)} := \mathrm{Norm}\left[[q^{(k-1)} - p^{(k-1)}]^+\right], k = 2, ..., K+1$. If all draft tokens are rejected, we sample a token $Y \sim q^{(K+1)}$ from the last residual distribution. Due to the limited pages, we remain other related works in Appendix A.

3 RECURSIVE SPECULATIVE DECODING

In this section, we present **R**ecursive **S**peculative **D**ecoding (RSD), a tree-based speculative decoding method that constructs draft-token trees via sampling without replacement. We first propose recursive rejection sampling that generalizes multi-round speculative decoding (Miao et al., 2023) and is applicable to draft distributions with dependencies, where sampling-without-replacement distribution is one instance of such distributions. Then, we use recursive rejection sampling to validate each level of the draft-token tree which can be efficiently constructed via either Gumbel-Top-k trick (Vieira, 2014) and Stochastic Beam Search (Kool et al., 2019),

3.1 RECURSIVE REJECTION SAMPLING: GENERALIZED MULTI-ROUND REJECTION SAMPLING

Suppose we have target distribution $q(x), x \in \mathcal{X}$. In recursive rejection sampling, we introduce random variables $\hat{X}^{(1)}, ..., \hat{X}^{(K)} \in \mathcal{X}$ that represent K draft tokens; these tokens will locate at the same level of the draft-token tree in Section 3.2. We aim to recover target distribution q, where

$$\hat{X}^{(1)} \sim p^{(1)}, \hat{X}^{(k)} \sim p^{(k)}(\cdot|\hat{X}^{(1:k-1)}), \quad k = 2, ..., K, \tag{1}$$

for some distributions $p^{(k)}, k = 1, ..., K$ and a sequence $\hat{X}^{(1:k-1)} := (\hat{X}^{(1)}, ..., \hat{X}^{(k-1)})$. Note that we assume distributions with dependencies unlike prior works such as SpecTr Sun et al. (2023) consider independent distributions. By using $p^{(1)}, ..., p^{(K)}$ and q, we define $q^{(1)} := q$ and residual distributions

Algorithm 1 Recursive Rejection Sampling

1: **Input:** Draft dist. $p^{(k)}, k = 1, ..., K$, target dist. q.
2: Sample $\hat{X}^{(k)}$ by equation 1.
3: Compute $q^{(k)}(\cdot|\hat{X}^{(1:k-2)})$ and $\Theta^{(k)}$ by equation 2 and equation 3.
4: **for** k **in** $\{1, ..., K\}$ **do**
5: Sample $A^{(k)} \in \{\texttt{acc}, \texttt{rej}\}$ with probability $\Theta^{(k)}$.
6: **if** $A^{(k)} = \texttt{acc}$ **then return** $Z \leftarrow \hat{X}^{(k)}$; **end if**
7: **end for**
8: **return** $Z \sim q^{(K+1)}(\cdot|\hat{X}^{(1:K-1)})$

$$q^{(k+1)}(\cdot|x^{(1:k-1)}) := \mathrm{Norm}\left[[q^{(k)}(\cdot|x^{(1:k-2)}) - p^{(k)}(\cdot|x^{(1:k-1)})]^+\right] \tag{2}$$

for $k = 1, ..., K$ and $x^{(1)}, ..., x^{(K+1)} \in \mathcal{X}$, where $x^{(1:k')} = \emptyset$ (empty sequence, i.e., no conditioning) if $k' < 1$, or $(x^{(1)}, ..., x^{(k')})$, otherwise. Together with draft, target, and residual distributions, recursive rejection sampling introduces threshold random variables $\Theta^{(1)}, ..., \Theta^{(K)} \in [0, 1]$ which determines rejection criteria for each draft token $\hat{X}^{(k)}, k = 1, ..., K$:

$$\Theta^{(k)} := \min \left\{ 1, \frac{q^{(k)}(\hat{X}^{(k)} | \hat{X}^{(1:k-2)})}{p^{(k)}(\hat{X}^{(k)} | \hat{X}^{(1:k-1)})} \right\}. \tag{3}$$

Specifically, each $\Theta^{(k)}$ can be used to define random variables $A^{(k)} \in \{\texttt{acc}, \texttt{rej}\}$ (where `acc` and `rej` indicate acceptance and rejection of draft tokens, respectively) such that $\Pr\left\{A^{(k)} = \texttt{acc} | \Theta^{(k)} = \theta\right\} = \theta$ for $\theta \in [0, 1]$.

Finally, recursive rejection sampling can be characterized by defining a random variable $Z \in \mathcal{X}$ such that

$$Z := \begin{cases} \hat{X}^{(k)}, & \text{if } A^{(1:k-1)} = \texttt{rej}^{k-1}, A^{(k)} = \texttt{acc}, \\ & \quad k = 1, ..., K, \\ Y, & \text{if } A^{(1:K)} = \texttt{rej}^{K}, \\ & \quad Y \sim q^{(K+1)}(\cdot | \hat{X}^{(1:K-1)}), \end{cases} \tag{4}$$

where $A^{(1:k-1)} := (A^{(1)}, ..., A^{(k-1)})$ and $\texttt{rej}^k$ is a length-k sequence with all of its elements equal to `rej`. Intuitively, we select $\hat{X}^{(1)}$ if it is accepted ($A^{(1)} = \texttt{acc}$); we select $\hat{X}^{(k)}$ when all previous draft tokens $\hat{X}^{(1)}, ..., \hat{X}^{(k-1)}$ are rejected *and* $\hat{X}^{(k)}$ is accepted ($A^{(1:k-1)} = \texttt{rej}^{k-1}, A^{(k)} = \texttt{acc}$) for each k; we sample $Y \sim q^{(K+1)}(\cdot | \hat{X}^{(1:K-1)})$ and select Y if all draft tokens are rejected ($A^{(1:K)} = \texttt{rej}^K$). We summarize the entire process of recursive rejection sampling in **Algorithm 1**. Note that the original rejection sampling Leviathan et al. (2023); Chen et al. (2023) is a special case of our recursive rejection sampling with $K = 1$. Also, it can be shown that recursive rejection sampling equation 4 always recovers the target distribution q:

Theorem 3.1 (Recursive rejection sampling recovers target distribution)**.** *For the random variable $Z \in \mathcal{X}$ in equation 4,* $\Pr\{Z = z\} = q(z), z \in \mathcal{X}$. *(See Appendix B.1 for the proof.)*

Although the proposed recursive rejection sampling is applicable to arbitrary distributions with dependencies following equation 1, we assume a single draft model (as in SpecTr (Sun et al., 2023) and focus on the cases where the draft model samples predictive tokens without replacement, which is an instance of equation 1.

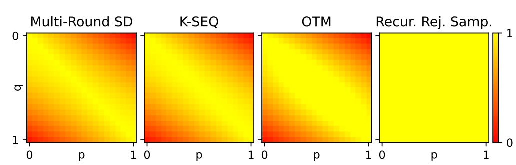


Figure 1: Acceptance rates for multi-round speculative decoding, K-SEQ, OTM and recursive rejection sampling are given when $\mathrm{Ber}(p)$ and $\mathrm{Ber}(q)$ are draft and target distributions, respectively, and two tokens are proposed by the draft model ($K = 2$).

Toy example. We present a didactic example with Bernoulli distributions (given by Sun et al. (2023)) to showcase the benefit of recursive rejection sampling. Suppose that Bernoulli distributions are used for both draft and target models and only $K = 2$ tokens are allowed for draft proposals. The acceptance rates for different methods are depicted in Figure 1; multi-round speculative decoding (from SpecInfer (Miao et al., 2023)), K-SEQ and Optimal Transport with Membership costs (OTM) (Sun et al., 2023), use sampling *with* replacement, whereas recursive rejection sampling uses sampling *without* replacement; note that both K-SEQ and OTM were presented in SpecTr paper (Sun et al., 2023) where OTM shows theoretically optimal acceptance rate. For all the baselines, acceptance rates decrease as the discrepancy between draft and target distribution increases, since tokens sampled from draft models become more unlikely from target models. On the other hand, recursive rejection sampling achieves 100% acceptance rate even with high draft-target-model discrepancy; once the first draft token is rejected, the second draft token is always aligned with the residual distribution. This example shows that draft distributions with dependencies, e.g., sampling-without-replacement distribution, leads to higher acceptance rate and becomes crucial, especially for the cases with higher distributional discrepancy between draft and target.

3.2 Tree-Based Speculative Decoding with Recursive Rejection Sampling

Recursive rejection sampling is applicable to tree-based speculative decoding algorithms if sampling without replacement is used to construct a *draft-token tree*. Two Recursive Speculative Decoding

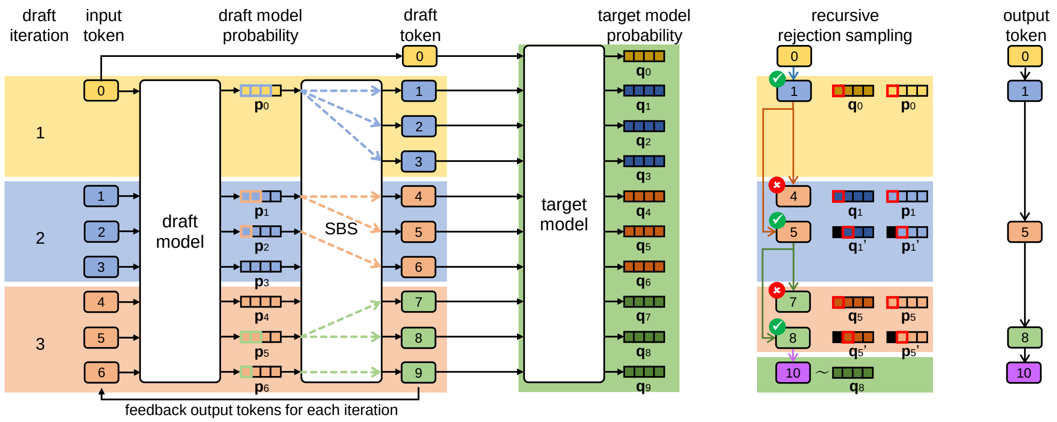
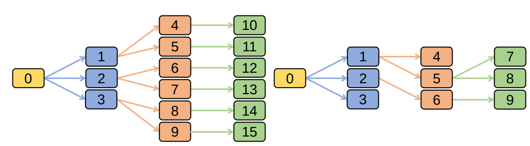


Figure 2: We describe the entire process of RSD with Stochastic Beam Search (RSD-S); the difference between RSD-S and RSD with Constant branching factors (RSD-C) lies at the method of constructing the draft-token tree. Draft tokens the tree are sampled in parallel at each level and auto-regressively across levels, while Stochastic Beam Search samples sequences without replacement at each tree level. The established draft-token tree is then processed by the target model in parallel, which lets us acquire the token-wise target model probabilities. Finally, recursive rejection sampling (for sampling-without-replacement distribution) is applied to each level of the tree, recovering the sequence generation distribution of the target model.

(RSD) algorithms using recursive rejection sampling are presented in this section, while they share the same pipeline for parallel target evaluation and draft tree verification after building the draft-token tree (See *Figure 2.*). We describe details about how RSD works in the following sections.

3.2.1 DRAFT-TOKEN TREE GENERATION

We consider two RSD algorithms: **RSD** with **C**onstant branching factors (RSD-C) and **RSD** with **S**tochastic Beam Search (RSD-S). RSD-C builds the draft-token tree having constant branching factors, which makes sequences from the tree to have the same length. RSD-S, on the other hand, builds the tree via Stochastic Beam Search Kool et al. (2019) that samples *draft sequences* without replacement, while truncating sequences that are unlikely to be generated from the draft model and efficiently handling the computational cost.



(a) RSD-C *(b)* RSD-S

Figure 3: We describe examples of constructing draft-token trees with the (maximum) draft length equal to 3; *(a)* The tree constructed by RSD-C with branching factors $\mathbf{b} = (3, 2, 1)$ is given; *(b)* we depict the tree constructed by RSD-S with beamwidth $W = 3$, where edges are determined via Stochastic Beam Search.

RSD with Constant branching factors (RSD-C). Let L denote the fixed length for all draft sequences, which is equivalent to the depth of the draft-token tree, and $\boldsymbol{\tau}_0^{(1)}$ denote the input sequence of tokens. Let us assume that the tree level increases from root ($l = 0$) to leaf ($l = L$) nodes, where each node is characterized by the (partial) sequence. We also define $\mathbf{b} := (b_0, ..., b_{L-1})$ where b_l is the branching factor at the level l (See *Figure 3(a)* for the example with $\mathbf{b} = (3, 2, 1)$.).

At each level $l \in \{0, ..., L-1\}$ of the draft tree, we begin with N_l sequences $\boldsymbol{\tau}_l^{(k)}, k = 1, ..., N_l$ generated from the previous level, where $N_0 := 1$ and $N_l := \prod_{l'=0}^{l-1} b_{l'}$ for $l \geq 1$. Then, we evaluate log probabilities $\phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$ and perturbed log probabilities $\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$ for each k, i.e., for i.i.d. Gumbel samples $G_l^{(k)}(x), x \in \mathcal{X}$,

$$\phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot) := \log p(\cdot|\boldsymbol{\tau}_l^{(k)}), \tag{5}$$
$$\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot) := \phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot) + G_l^{(k)}, \tag{6}$$

where both log probabilities and Gumbel samples can be computed in parallel; proper positional encodings and attention masking (Cai et al., 2023; Miao et al., 2023) are required for the parallel log-probability computation when transformer architecture is used (Vaswani et al., 2017). By using *Gumbel-Top-k trick* (Vieira, 2014; Kool et al., 2019) with perturbed log probabilities equation 6, one can sample top-b_l tokens without replacement for each sequence $\tau_l^{(k)}$:

$$\hat{X}_{l+1}^{((k-1)b_l+1)}, ..., \hat{X}_{l+1}^{((k-1)b_l+b_l)} = \underset{x\in\mathcal{X}}{\text{argtop-}b_l} \left(\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, x)\right). \tag{7}$$

Note that the outputs $\hat{X}_{l+1}^{((k-1)b_l+k')}, k' = 1, ..., b_l$, in equation 7 are assumed to be in the decreasing order of values $\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \hat{X}_{l+1}^{((k-1)b_l+k')})$, for each k. Finally, we define

$$O_{l+1}^{((k-1)b_l+k')} := (\hat{X}_{l+1}^{((k-1)b_l+k')}, k), \boldsymbol{\tau}_{l+1}^{((k-1)b_l+1)} := (\boldsymbol{\tau}_l^{(k)}, \hat{X}_{l+1}^{((k-1)b_l+1)}) \tag{8}$$

for $k \in 1, ..., N_l$ and $k' \in \{1, ..., b_l\}$, where $O_{l+1}^{((k-1)b_l+k')}$ is a pair of draft token and parent sequence index. Those pairs in equation 8 are stored for all levels $l = 0, ..., L-1$ and used for draft tree verification, which exploits the fact that *the tokens $\hat{X}_{l+1}^{((k-1)b_l+1)}, ..., \hat{X}_{l+1}^{((k-1)b_l+b_l)}$ follow sampling without replacement from $p(\cdot|\boldsymbol{\tau}_l^{(k)})$ for any given parent sequence index k.*

RSD with Stochastic Beam Search (RSD-S). One caveat of RSD-C is that its constant branching factors $\mathbf{b}$ should be carefully determined to handle tree complexity, when the computation budget is limited; for example, if $\mathbf{b} = (n, ..., n)$ with its length L, the number of nodes in the draft tree will be $\sum_{l=0}^{L-1} n^l = O(n^{L-1})$, which is computationally prohibitive for large n and L. Also, RSD-C constructs sequences at each level l by using the *myopic* token-wise log probabilities ϕ_l in equation 6. RSD-S addresses both issues by using Stochastic Beam Search (Kool et al., 2019) that early-truncates unlikely sequences and utilizes *far-sighted* sequence log probabilities.

Let us define the *maximum* draft sequence length L and the beamwidth W. We also define $\boldsymbol{\tau}_0^{(1)}$ as the input sequence similar to RSD-C. At each level $l \in \{0, ..., L-1\}$, SBS uses *beam*

$$\mathcal{B}_l := (t_l^{(1)}, ..., t_l^{(W)}), t_l^{(k)} := (\boldsymbol{\tau}_l^{(k)}, \phi_{l-1}(\boldsymbol{\tau}_l^{(k)}), \psi_{l-1}(\boldsymbol{\tau}_l^{(k)}))$$

generated from the previous level $l-1$[1]. Here, each tuple $t_l^{(k)}$ for $k \in \{1, ..., W\}$ consists of *(a)* a sequence $\boldsymbol{\tau}_l^{(k)}$, *(b)* its *sequence* log probability $\phi_{l-1}(\boldsymbol{\tau}_l^{(k)})$ of $\boldsymbol{\tau}_l^{(k)}$, and *(c)* the *transformed (perturbed and truncated)* sequence log probability $\psi_{l-1}(\boldsymbol{\tau}_l^{(k)})$, respectively.

For each tuple $t_l^{(k)}$ in the beam $\mathcal{B}_l$, we evaluate the (next-level) sequence log probabilities $\phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$ and the perturbed sequence log probabilities $\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$. Specifically for i.i.d. Gumbel samples $G_l^{(k)}(x), x \in \mathcal{X}$, we compute

$$\phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot) := \phi_{l-1}(\boldsymbol{\tau}_l^{(k)}) + \log p(\cdot|\boldsymbol{\tau}_l^{(k)}), \tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot) := \phi_l(\boldsymbol{\tau}_l^{(k)}, \cdot) + G_l^{(k)},$$

where the terms $\boldsymbol{\tau}_l^{(k)}$ and $\phi_{l-1}(\boldsymbol{\tau}_l^{(k)})$ within the tuple $t_l^{(k)}$ of within the beam $\mathcal{B}_l$ are reused. Similar to RSD-C, both log probabilities and Gumbel samples can be parallelly computed with positional encodings and attention masking (Cai et al., 2023; Miao et al., 2023). In addition to the perturbed log probabilities, SBS in RSD-S transforms $\tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$ into the truncated function

$$\psi_l(\boldsymbol{\tau}_l^{(k)}, \cdot) := T(\psi_{l-1}(\boldsymbol{\tau}_l^{(k)}), \tilde{\phi}_l(\boldsymbol{\tau}_l^{(k)}, \cdot)), \tag{9}$$

$$T(u, \phi) := -\log\left(e^{-u} - e^{-\max\phi} + e^{-\phi(\cdot)}\right) \tag{10}$$

for $\max\phi := \max_{x\in\mathcal{X}} \phi(x)$ by reusing $\psi_{l-1}(\boldsymbol{\tau}_l^{(k)})$ in $t_l^{(k)}$. Note that $T(u, \phi)$ in equation 10 is *monotonically increasing* w.r.t. ϕ and transforms ϕ to the function with the upper bound u (Kool et al., 2019)[2]

[1] For $l = 0$, $\phi_{-1}(\boldsymbol{\tau}_0^{(1)}) = \phi_{-1}(\boldsymbol{\tau}_0^{(1)}) = 0$ is used with $\mathcal{B}_0 := (t_0^{(1)})$ (Kool et al., 2019).

[2] In Appendix B.3 of Kool et al. (2019), a numerical stable way of evaluating the function T in equation 10 is provided.

After evaluating $\psi_l(\boldsymbol{\tau}_l^{(k)}, \cdot)$ for all parent sequences $\boldsymbol{\tau}_l^{(k)}$s, SBS selects top-$W$ pairs $(\hat{X}_{l+1}, p_{l+1})$ of draft token and parent sequence index *across the beam* $\mathcal{B}_l$, i.e.,

$$O_{l+1}^{(1)}, ..., O_{l+1}^{(W)} := \underset{(x,k)\in\mathcal{X}\times\mathcal{K}}{\text{argtop-}W} \left(\psi_l(\boldsymbol{\tau}_l^{(k)}, x)\right) \tag{11}$$

for $O_{l+1}^{(k)} := (\hat{X}_{l+1}^{(k)}, p_{l+1}^{(k)})$ and $\mathcal{K} := \{1, ..., W\}$. The output pairs $O_{l+1}^{(1)}, ..., O_{l+1}^{(W)}$ are given by *corresponding values* $\psi_l(\boldsymbol{\tau}_l^{(k)}, \hat{X}_{l+1}^{(k)})$ *in the decreasing order*. Finally, we construct the next beam

$$\mathcal{B}_{l+1} := (t_{l+1}^{(1)}, ..., t_{l+1}^{(W)}), t_{l+1}^{(k)} := ((\hat{\boldsymbol{\tau}}_{l+1}^{(k)}, \hat{X}_{l+1}^{(k)}), \phi_l(\hat{\boldsymbol{\tau}}_{l+1}^{(k)}, \hat{X}_{l+1}^{(k)}), \psi_l(\hat{\boldsymbol{\tau}}_{l+1}^{(k)}, \hat{X}_{l+1}^{(k)}))$$

for $k = 1, ..., W$, where $\hat{\boldsymbol{\tau}}_{l+1}^{(k)} := \boldsymbol{\tau}_l^{(p_{l+1}^{(k)})}$ is the selected parent sequence. Intuitively, SBS at the level l evaluates scores $\psi_l^{(k)}(\boldsymbol{\tau}_l^{(k)}, x), x \in \mathcal{X}, k \in \mathcal{K}$, by considering all child nodes from the beam $\mathcal{B}_l$. SBS selects W nodes among all child nodes having top-W scores. Note that the above process is theoretically equivalent to sample top-W length-$(l+1)$ sequences *without replacement* (Kool et al., 2019) and efficiently truncates sequences that are unlikely to be generated. (See *Figure 3(b)*.)

We store the *ordered* sequence of pairs $O_{l+1}^{(1)}, ..., O_{l+1}^{(W)}$ for all levels $l = 0, ..., L-1$, which is used for draft-tree verification. As in RSD-C, we show the following property:

Theorem 3.2 (Tokens from the same sequence follow sampling without replacement in RSD-S)**.** *In RSD-S, any non-empty subsequence of the sequence $\hat{X}_{l+1}^{(1)}, ..., \hat{X}_{l+1}^{(W)}$ of draft tokens (from $O_{l+1}^{(1)}, ..., O_{l+1}^{(W)}$ in equation 11) such that each element of the subsequence has the same parent $\boldsymbol{\tau}_l^{(k)}$ follows sampling without replacement from $p(\cdot|\boldsymbol{\tau}_l^{(k)})$[3]. See Appendix B.2 of the proof.*

3.2.2 Draft-tree evaluation and verification

Tree evaluation with target model. After the draft-tree construction, we have sequences of pairs $(O_l^{(1)}, ..., O_l^{(N_l)}), l = 1, ..., L$, where $N_l = \prod_{l'=0}^{l} b_{l'}$ for RSD-C and $N_l = W$ for RSD-S, respectively ($N_0 := 1$ for both). Those pairs include the node-connection information of the draft tree and can be used to *parallelly* evaluate the draft tree via the target model by utilizing appropriate attention masking and positional encodings. From the evaluation process, we acquire the target log probabilities for all sequences $\boldsymbol{\tau}_l^{(k_l)}$ in the draft tree, i.e., $q(\cdot|\boldsymbol{\tau}_l^{(k_l)}), l = 0, ..., L, k_l = 1, ..., N_l$.

Verification via recursive rejection sampling. Earlier, we show that tokens in the tree having the same parent sequence $\boldsymbol{\tau}_l^{(k_l)}$ follows the sampling-without-replacement distribution from $p(\cdot|\boldsymbol{\tau}_l^{(k_l)})$ for both RSD-C and RSD-S. Thus, one can apply recursive rejection sampling iteratively at each tree level. Specifically, at the level $l \in \{0, 1, ..., L\}$, we begin with a sequence $\boldsymbol{\tau}_l^{(k_l')}$ where k_l' is the index of the parent sequence accepted in the previous level ($k_0' = 1$ at the level $l = 0$). Within the *ordered* sequence $(O_{l+1}^{(1)}, ..., O_{l+1}^{(N_{l+1})})$ of pairs, we find the subsequence $\mathbf{o}_{l+1}^{(k_l')}$ having $\boldsymbol{\tau}_l^{(k_l')}$ as parent, which can be validated by checking the second element of each pair $O_{l+1}^{(k)}$, and the token sequence $\mathbf{x}_{l+1}^{(k_l')}$ in $\mathbf{o}_{l+1}^{(k_l')}$. Earlier, we show that tokens $\mathbf{x}_{l+1}^{(k_l')}$ follows sampling-without-replacement distribution in its order, so we can apply recursive rejection sampling to those tokens with draft and target distributions, $p(\cdot|\boldsymbol{\tau}_l^{(k_l')})$ and $q(\cdot|\boldsymbol{\tau}_l^{(k_l')})$, respectively. If any token x in $\mathbf{x}_{l+1}^{(k_l')}$ is accepted, we set k_{l+1}' that corresponds to $\boldsymbol{\tau}_l^{(k_{l+1}')} := (\boldsymbol{\tau}_l^{(k_l')}, x)$, and we continue to the next-level verification if child nodes exist. If all tokens are rejected, we sample from the last residual distribution of recursiver rejection sampling. If there is no child node, we sample from the target $q(\cdot|\boldsymbol{\tau}_l^{(k_l')})$ similar to the single-sequence speculative decoding (Chen et al., 2023; Leviathan et al., 2023). We provide detailed descriptions for RSD-C (**Algorithm 2**) and for RSD-S (**Algorithm 7**) in Appendix C.

4 Experiments

We evaluate RSD-C and RSD-S together with our baselines including speculative decoding (SD) (Chen et al., 2023; Leviathan et al., 2023) and SpecTr (Sun et al., 2023), where a single draft model is assumed[4]. We consider the following perspectives during our experiments: ***(Exp1)*** How

[3]We define a subsequence of a sequence as any sequence acquired by removing its elements *while maintaining the order in the original sequence*.

[4]We exclude SpecInfer (Miao et al., 2023) from our baselines since it uses multiple draft models.

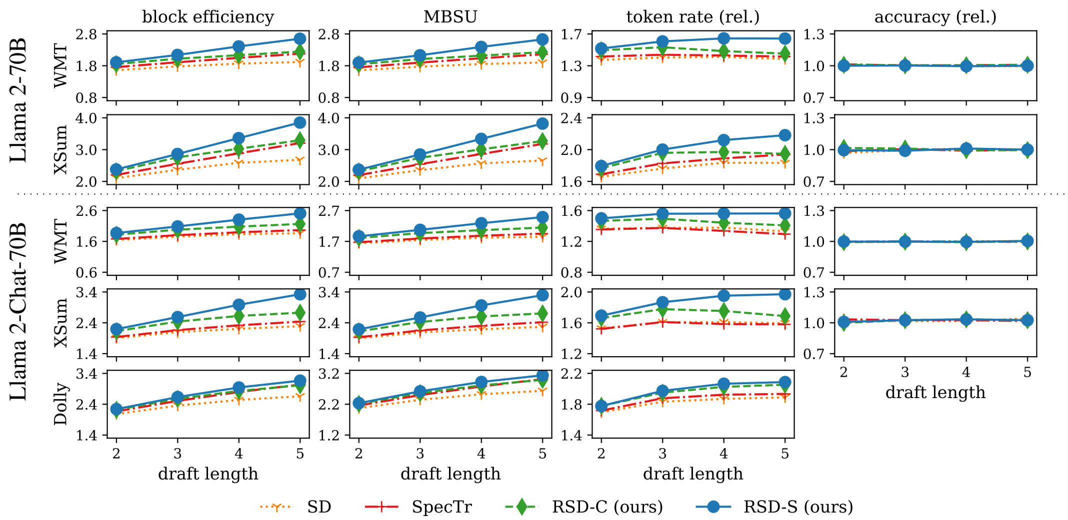


Figure 4: Block efficiency, MBSU, token rate and accuracy for various lengths $(2, 3, 4, 5)$ of draft sequences are given. We consider two target models, Llama 2-70B and Llama 2-Chat-70B, each of which has a corresponding smaller draft model for speculative decoding. All results are normalized by the corresponding numbers from auto-regressive decoding. RSD-S always outperforms SD, SpecTr and RSD-C. All methods including auto-regressive decoding show similar accuracy for WMT and XSum.

will the performance be affected by *the length of draft sequences*? ***(Exp2)*** How will the performance be affected by *the target computational budget*, i.e., the number of tokens processed at the target model? While ***(Exp1)*** has been frequently investigated by existing tree-based speculative decoding methods (Sun et al., 2023; Miao et al., 2023), ***(Exp2)*** has not been conducted as far as we concern, which has practical importance when running the target model on resource-bounded devices.

Models. We consider the following target models; **Llama 2** and **Llama 2-Chat** (Touvron et al., 2023) with **7B**, **13B** and **70B** parameters; **OPT** (Zhang et al., 2022) with **13B**, **30B** and **66B** parameters. Each class of target models adopts corresponding draft model; see Appendix D.1. In this section, we only present Llama 2-70B and Llama 2-Chat-70B results, and other results (Llama 2 with other sizes and OPT) can be found in Appendix D.4.

Tasks. Our methods and baselines are evaluated for **WMT**18-DeEn (Bojar et al., 2018, translation) and **XSum** (Narayan et al., 2018, summarization) for each target model, while we report accuracy scores (BLEU for WMT and ROUGE-2 for XSum) to confirm if the target model's distribution is recovered; Databricks-**Dolly**-15k (Conover et al., 2023, question and answering) is used only for Llama 2-Chat without accuracy evaluation. We use temperature 0.3 for both XSum and WMT and 1.0 for Dolly, where we further apply nucleus (top-p) sampling (Holtzman et al., 2019) with $p = 0.95$ for Dolly.

Performance metrics. We evaluate **block efficiency** (Leviathan et al., 2023), Memory-Bound Speed-Up (**MBSU**) (Zhou et al., 2023) and **token rate** (tokens/sec) on A100 GPUs; see Appendix D.2 for details.

4.1 *(Exp 1)* FIXED DRAFT SEQUENCE LENGTH

We fix (maximum) draft sequence length as the value in $\{2, 3, 4, 5\}$ and evaluate our methods and baselines, which is summarized in *Figure 4*. Regarding the tree structures of each decoding methods, we let both SpecTr and RSD-S always use draft-token trees, the size of which is smaller than or equal to that of RSD-C's tree; see Appendix D.3.1 for details. Our results show that tree-based methods (SpecTr, RSD-C and RSD-S) always outperform SD in terms of block efficiency and MBSU, whereas token rates for SpecTr and RSD-C can be lower than that for SD; this is since block efficiencies for both SpecTr and RSD-C are relatively low and there is additional computational overhead to process the tree. On the other hand, *RSD-S strictly outperforms both SD and SpecTr for all performance metrics*, showing the superiority of RSD-S over our baselines and the importance of early-truncating unlikely draft sequences. We also observe that there is no strong correlation between

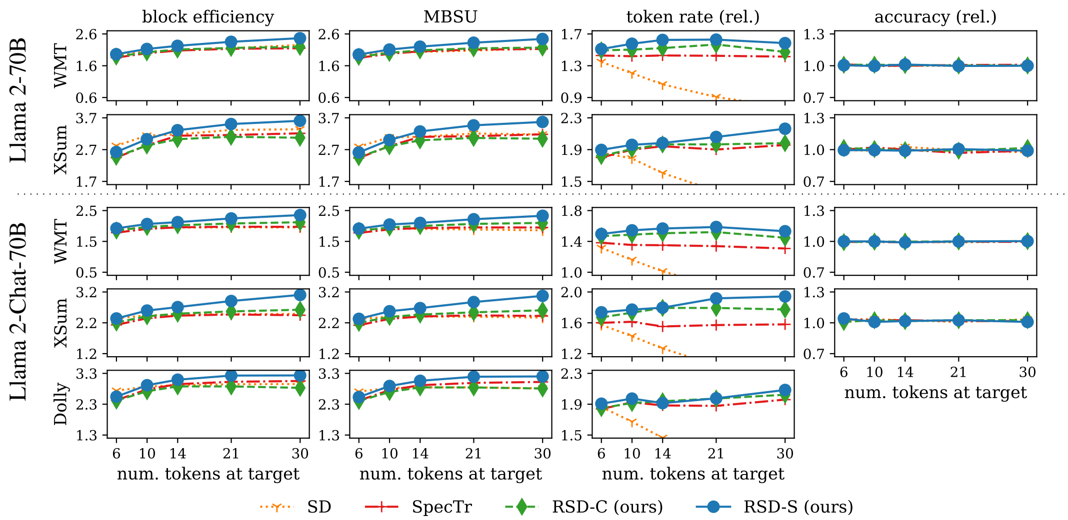


Figure 5: Block efficiency, MBSU, token rate and accuracy for various target computational budgets (the numbers $6, 10, 14, 21, 30$ of draft tokens processed at the target model) are given. We consider two target models, Llama 2-70B and Llama 2-Chat-70B, each of which has a corresponding smaller draft model for speculative decoding. All results are normalized by the corresponding numbers from auto-regressive decoding. RSD-S outperforms SD, SpecTr and RSD-C in the majority of cases. All methods including auto-regressive decoding show similar accuracy for both WMT and XSum.

MBSU and token rate; this is since A100 GPUs used to measure token rates are *not* memory-bound. Furthermore, token rates in many cases are shown to decrease as the length of draft-token sequence becomes higher, which is due to the increased computation overhead to execute draft models with the longer draft sequence; however, one needs to be cautious since this result may not generally hold since token rate is hugely affected by the efficiency of software implementation and the devices which we execute the methods on. Finally, in WMT and XSum, BLEU and ROUGE-2 scores are similar across different methods, respectively, which implies that all methods recover the distributions of target LLMs.

4.2 (***Exp2***) Fixed target computational budget

We select target computational budget, i.e., the number of draft tokens processed at the target model in parallel for each speculative decoding iteration, among values in $\{6, 10, 14, 21, 30\}$ and evaluate our proposed methods and baselines; we summarize the results in *Figure 5* and describe tree structures in Appendix D.3.2. While RSD-S achieves higher block efficiency and MBSU than SD and SpecTr in most cases, SD beats RSD-C in the relatively low budget regime, e.g., $\{6, 10\}$ with Llama 2-70B and XSum, and $\{6\}$ with Llama 2-Chat-70B and Dolly. We believe that our draft models are well-aligned with corresponding target models for those cases (from the observation that block efficiencies of SD close to 3.0, which are significantly higher than the numbers in other cases, are achieved), and increasing the depth rather than the width of the tree could quickly increase the acceptance rate in such cases. In the high budget regime, on the other hand, RSD-S beats SD for both block efficiency and MBSU. In terms of token rate, RSD-S strictly outperforms our baselines, whereas SD's token rate severely decreases for higher target computation budgets due to the computational overhead caused by the draft model's auto-regressive decoding with the longer draft sequence.

5 Conclusion

We present RSD algorithms, a novel tree-based speculative decoding method leveraging the full diversifiability of the draft-token tree; RSD-C efficiently samples draft tokens without replacement via Gumbel-Top-k trick, while RSD-S uses Stochastic Beam Search and samples draft-token sequences without replacement. We also propose recursive rejection sampling that can verify the tree built by the sampling-without-replacement process and recovers the exact target model distribution. We show that RSD outperforms the baselines in most cases, supporting the importance of diverse drafting when accelerating LLM inference.

REFERENCES

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

Ondrej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pp. 272–307, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W18-6401`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901, 2020.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. Medusa: Simple framework for accelerating llm generation with multiple decoding heads. `https://github.com/FasterDecoding/Medusa`, 2023.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

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

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL `https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm`.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Breaking the sequential dependency of LLM inference using lookahead decoding, November 2023. URL `https://lmsys.org/blog/2023-11-21-lookahead-decoding/`.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*, 2023.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-Top-k trick for sampling sequences without replacement. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 3499–3508. PMLR, 2019.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating generative LLM serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

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

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Tim Vieira. Gumbel-max trick and weighted reservoir sampling. 2014. URL `https://timvieira.github.io/blog/post/2014/08/01/gumbel-max-trick-andweighted-reservoir-sampling/`.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

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

A Related Works

Many recent works have aimed to address the inference bottleneck of LLMs caused by auto-regressive decoding. Speculative decoding methods (Leviathan et al., 2023; Chen et al., 2023; Sun et al., 2023; Miao et al., 2023) use the target model (LLM) with a draft model (a small language model), while recovering target distribution via rejection sampling. See the recent survey on speculative decoding (Xia et al., 2024) for more comprehensive understanding.

Other than speculative decoding methods, BiLD (Kim et al., 2023) is another method to accelerate inference, where it uses a fallback policy which determines when to invoke the target model and a rollback policy to review and correct draft tokens. Medusa (Cai et al., 2024) uses multiple decoding heads to predict future tokens in parallel, constructs the draft-token tree and uses a typical acceptance criteria. Lookahead decoding Fu et al. (2023) caches the historical n-grams generated on-the-fly instead of having a draft model and performs parallel decoding using Jacobi iteration and verifies n-grams from the cache. While showing promising results with greedy sampling, these works do not guarantee target distribution recovery in contrast to speculative decoding methods.

B Theorems and proofs

B.1 Proof of **Theorem 3.1**

Theorem 3.1 (Recursive rejection sampling recovers target distribution)**.** *The random variable $Z \in \mathcal{X}$ defining recursive rejection sampling rule equation 4 follows the target distribution q, i.e.,*

$$\Pr\{Z = z\} = q(z), z \in \mathcal{X}.$$

Proof. We remain a sketch of the proof here and the formal proof is given in the next paragraph. We first consider the case where $\hat{X}^{(1)}, ..., \hat{X}^{(K-1)}$ are rejected and see whether we accept $\hat{X}^{(K)}$ or not; we either accept $\hat{X}^{(K)}$ with probability $\Theta^{(K)}$ in equation 3 or sample a new token $Y \sim q^{(K+1)}(\cdot|\hat{X}^{(1:K-1)})$ when all draft tokens are rejected. Since $q^{(K+1)}$ is the residual distribution from $q^{(K)}$, one can regard it as the simple sampling by Chen et al. (2023) and Leviathan et al. (2023), which recovers $q^{(K)}$. The same idea is applied to $\hat{X}^{(K-1)}, ..., \hat{X}^{(1)}$ in the reversed order until we recover $q = q^{(1)}$ at the end.

Let us desribe the formal proof. From the definition of recursive rejection sampling equation 4, we have

$$\begin{aligned}
&\Pr\{Z = z\} \\
&= \sum_{k=1}^{K} \underbrace{\Pr\left\{A^{(1:k-1)} = \texttt{rej}^{k-1}, \hat{X}^{(k)} = z, A^{(k)} = \texttt{acc}\right\}}_{=:\Sigma_{1,k}} + \underbrace{\Pr\left\{A^{(1:K)} = \texttt{rej}^{K}, \tilde{X}^{(K+1)} = z\right\}}_{=:\Sigma_{2,K}}.
\end{aligned} \tag{12}$$

It can be shown that the following equality holds for each k:

$$\Sigma_{2,k-1} = \Sigma_{1,k} + \Sigma_{2,k}. \tag{13}$$

Let us first consider $k = K$, then,

$$\begin{aligned}
&\Sigma_{1,K} + \Sigma_{2,K} \\
&= \sum_{x^{(1)},\dots,x^{(K-1)}} \Pr\left\{\hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\times \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1}, \hat{X}^{(K)} = z, A^{(K)} = \texttt{acc} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&+ \sum_{x^{(1)},\dots,x^{(K)}} \Pr\left\{\hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&\times \Pr\left\{A^{(1:K)} = \texttt{rej}^{K}, \hat{X}^{(K+1)} = z \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&= \sum_{x^{(1)},\dots,x^{(K-1)}} \Pr\left\{\hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \Bigg(\\
&\underbrace{\Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1}, \hat{X}^{(K)} = z, A^{(K)} = \texttt{acc} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\}}_{=:T_1(K)} \\
&\underbrace{+ \sum_{x^{(K)}} \Pr\left\{\hat{X}^{(K)} = x^{(K)} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \times \Pr\left\{A^{(1:K)} = \texttt{rej}^{K}, \hat{X}^{(K+1)} = z \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\}}_{=:T_2(K)} \Bigg).
\end{aligned}$$

One can represent $T_{1,K}$ and $T_{2,K}$ as follows:

$$
\begin{aligned}
&T_{1,K} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\quad \times \Pr\left\{\hat{X}^{(K)} = z \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \times \Pr\left\{A^{(K)} = \texttt{acc} \middle| \hat{X}^{(1:K)} = (x^{(1:K-1)}, z)\right\} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} p^{(K)}(z|x^{(1:K-1)}) \min\left\{1, \frac{q^{(K)}(z|x^{(1:K-2)})}{p^{(K)}(z|x^{(1:K-1)})}\right\} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \min\left\{p^{(K)}(z|x^{(1:K-1)}), q^{(K)}(z|x^{(1:K-2)})\right\},
\end{aligned}
$$

$$
\begin{aligned}
&T_{2,K} \\
&= \sum_{x^{(K)}} \Pr\left\{\hat{X}^{(K)} = x^{(K)} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\times \Pr\left\{A^{(1:K)} = \texttt{rej}^{K}, \hat{X}^{(K+1)} = z \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&= \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \Pr\left\{A^{(1:K)} = \texttt{rej}^{K} \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&\times \Pr\left\{\hat{X}^{(K+1)} = z \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&= \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \Pr\left\{A^{(1:K)} = \texttt{rej}^{K} \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \times q^{(K+1)}(z|x^{(1:K-1)}) \\
&= \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\qquad \times \Pr\left\{A^{(K)} = \texttt{rej} \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \times q^{(K+1)}(z|x^{(1:K-1)}) \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} q^{(K+1)}(z|x^{(1:K-1)}) \\
&\qquad \times \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \Pr\left\{A^{(K)} = \texttt{rej} \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} q^{(K+1)}(z|x^{(1:K-1)}) \\
&\qquad \times \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \left(1 - \Pr\left\{A^{(K)} = \texttt{acc} \middle| \hat{X}^{(1:K)} = x^{(1:K)}\right\}\right) \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} q^{(K+1)}(z|x^{(1:K-1)}) \\
&\qquad \times \sum_{x^{(K)}} p^{(K)}(x^{(K)}|x^{(1:K-1)}) \times \left(1 - \min\left\{1, \frac{q^{(K)}(x^{(K)}|x^{(1:K-2)})}{p^{(K)}(x^{(K)}|x^{(1:K-1)})}\right\}\right) \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\qquad \times \frac{\max\left\{0, q^{(K)}(z|x^{(1:K-2)}) - p^{(K)}(z|x^{(1:K-1)})\right\}}{\sum_{x^{(K)}} \max\{0, q^{(K)}(x^{(k)}|x^{(1:K-2)}) - p^{(K)}(x^{(K)}|x^{(1:K-1)})\}} \\
&\qquad \times \sum_{x^{(K)}} \max\left\{0, q^{(K)}(x^{(K)}|x^{(1:K-2)}) - p^{(K)}(x^{(K)}|x^{(1:K-1)})\right\} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \max\left\{0, q^{(K)}(z|x^{(1:K-2)}) - p^{(K)}(z|x^{(1:K-1)})\right\}.
\end{aligned}
$$

Therefore, we have

$$
\begin{aligned}
&T_{1,K} + T_{2,K} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\qquad \times \left(\min\left\{p^{(K)}(z|x^{(1:K-1)}), q^{(K)}(z|x^{(1:K-2)})\right\} \right. \\
&\qquad \left. + \max\left\{0, q^{(K)}(z|x^{(1:K-2)}) - p^{(K)}(z|x^{(1:K-1)})\right\} \right) \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1} \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} q^{(K)}(z|x^{(1:K-2)}) \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1}, \tilde{X}^{(K)} = z \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\},
\end{aligned}
$$

where we define a random variable $\tilde{X}^{(K)}$ such that

$$
\Pr\left\{\tilde{X}^{(K)} = z \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} := q^{(K)}(z|x^{(1:K-1)}),
$$

which leads to

$$
\begin{aligned}
&\Sigma_{1,K} + \Sigma_{2,K} \\
&= \sum_{x^{(1)},\ldots,x^{(K-1)}} \Pr\left\{\hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} (T_{1,K} + T_{2,K}) \\
&= \sum_{x^{(1)},\ldots,x^{(K-1)}} \Pr\left\{\hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&\times \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1}, \tilde{X}^{(K)} = z \middle| \hat{X}^{(1:K-1)} = x^{(1:K-1)}\right\} \\
&= \Pr\left\{A^{(1:K-1)} = \texttt{rej}^{K-1}, \tilde{X}^{(K)} = z\right\} \\
&= \Sigma_{2,K-1}.
\end{aligned}
$$

Since the same derivation can be done for $k = 2, \ldots, K-1$, we have

$$
\Pr\{Z = z\} = \sum_{k=1}^{K} \Sigma_{1,k} + \Sigma_{2,K} = \sum_{k=1}^{K-1} \Sigma_{1,k} + \Sigma_{2,K-1} = \cdots = \Sigma_{1,1} + \Sigma_{2,1} = q(z),
$$

where the last equality holds from the derivation of original speculative decoding by (Chen et al., 2023; Leviathan et al., 2023). ☐

B.2 PROOF OF **THEOREM 3.2**

Theorem 3.2 (Tokens from the same sequence follow sampling without replacement in RSD-S)**.** *In RSD-S, any non-empty subsequence of the sequence $\hat{X}^{(1)}_{l+1}, \ldots, \hat{X}^{(W)}_{l+1}$ of draft tokens (from $O^{(1)}_{l+1}, \ldots, O^{(W)}_{l+1}$ in equation 11) such that each element of the subsequence has the same parent $\boldsymbol{\tau}^{(k)}_l$ follows sampling without replacement from $p(\cdot|\boldsymbol{\tau}^{(k)}_l)$.*

Proof. For fixed $\boldsymbol{\tau}^{(k)}_l$, consider a sequence of tokens

$$
\bar{\mathbf{X}}^{(k)}_{l+1} := \operatorname*{argsort}_{x\in\mathcal{X}} \psi_l(\boldsymbol{\tau}^{(k)}_l, x) = \operatorname*{argsort}_{x\in\mathcal{X}} \tilde{\phi}_l(\boldsymbol{\tau}^{(k)}_l, x),
$$

where the last equality holds since T in equation 9 is monotonically increasing w.r.t. $\tilde{\phi}_l(\boldsymbol{\tau}^{(k)}_l, \cdot)$ for fixed $\boldsymbol{\tau}^{(k)}_l$. Thus, $\bar{\mathbf{X}}^{(k)}_{l+1}$ can be seen as samples from $p(\cdot|\boldsymbol{\tau}^{(k)}_l)$ without replacement.

For a length-l_k subsequence $\mathbf{o}_{l+1}^{(k)}$ of $(O_{l+1}^{(1)}, ..., O_{l+1}^{(W)})$ in equation 11, where each element of the subsequence have $\boldsymbol{\tau}_l^{(k)}$ as its parent, the token sequence in $\mathbf{o}_{l+1}^{(k)}$ is a subsequence of $\bar{\mathbf{X}}_{l+1}^{(k)}$, i.e., those tokens are top-l_k samples without replacement from $p(\cdot|\boldsymbol{\tau}_l^{(k)})$. $\square$

C Algorithm

C.1 Recursive Speculative Decoding with Constant Branching Factors (RSD-C)

Algorithm 2 Recursive Speculative Decoding with Constant Branching Factors (RSD-C)

1: **Input:** The length L_{draft} of draft sequences (depth of the draft tree), a sequence $\mathbf{x}_{\text{input}}$ of input tokens, a list $\mathbf{b} := [b_0, ..., b_{L_{\text{draft}}-1}]$ of constant branching factors in the draft tree, the maximum length L_{output} of the output sequence.

2: `// Get the length of the input sequence.`
$L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$.

3: `// Initialize empty KV caches for draft and target models.`
$\mathbf{C}_{\text{draft}} \leftarrow \emptyset$, $\mathbf{C}_{\text{target}} \leftarrow \emptyset$.

4: **while** $L_{\text{input}} < L_{\text{output}}$ **do**

5: `// (STEP 1) Create a draft tree by using the draft model.`
$\mathcal{T}, \mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{M}, \mathbf{id}_{\text{position}}, \mathcal{L}_{\text{num\_nodes}} \leftarrow \texttt{CreateDraftTreeConst}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{b}, L_{\text{draft}})$.

6: `// (STEP 2) Evaluate draft tokens by using the target model.`
`// - Apply` $\mathbf{M}$ `to the right below corner of attention weights.`
`// - The target log probability` $\mathbf{\Phi}_{\text{target}}$ `is a` $\texttt{GetLength}(\mathbf{x}_{\text{input}}) \times N_{\text{vocab}}$ `tensor.`
`// -` N_{vocab} `is the vocabulary size.`
$\mathbf{\Phi}_{\text{target}}, \mathbf{C}_{\text{target}} \leftarrow \texttt{TargetModelForwardPass}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{target}}, \mathbf{id}_{\text{position}}, \mathbf{M})$.

7: `// - Convert the log probability tensor into the list of log probabilities for each level of the tree.`
$\mathcal{L}_{\text{log\_probs\_target}} \leftarrow \texttt{SplitTensor}(\mathbf{\Phi}_{\text{target}}[-\texttt{Sum}(\mathcal{L}_{\text{num\_nodes}}) :, :], \mathcal{L}_{\text{num\_nodes}}, \texttt{dim} = 0)$

8: `// (STEP 3) Run Recursive Rejection Sampling for each level of the tree.`
$\mathbf{x}_{\text{accepted}}, \mathbf{x}_{\text{last}}, \mathbf{id}_{\text{accepted\_flat\_node}} \leftarrow \texttt{RecursiveRejectionSampling}(\mathcal{T}, \mathcal{L}_{\text{log\_probs\_target}})$

9: `// (STEP 4) Use KV caches that are accepted, and prepare for the next round.`
$\mathbf{C}_{\text{draft}}, \mathbf{C}_{\text{target}} \leftarrow \texttt{FilterKVCache}(\mathbf{C}_{\text{draft}}, \mathbf{C}_{\text{target}}, L_{\text{input}}, \mathbf{id}_{\text{accepted\_flat\_node}})$

10: $\mathbf{x}_{\text{input}} \leftarrow \texttt{Concat}([\mathbf{x}_{\text{input}}[: L_{\text{input}}], \mathbf{x}_{\text{accepted}}, \mathbf{x}_{\text{last}}])$

11: $L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$

12: **end while**

13: **Output:** a sequence $\mathbf{x}_{\text{input}}$ that includes both input tokens and generated output tokens.

Algorithm 3 $\texttt{CreateDraftTreeConst}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{b}, L_{\text{draft}})$

1: **Input:** An input sequence $\mathbf{x}_{\text{input}}$, the draft KV cache $\mathbf{C}_{\text{draft}}$, the branching factor $\mathbf{b} := [b_0, ..., b_{L_{\text{draft}}-1}]$, the draft length L_{draft}

2: `// Get the length of the input sequence.`
$L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$.

3: `// Initialize lists for 1) draft log probabilities, 2) flattened node IDs, 3) parent node ids (within each level of the draft tree), 4) draft tokens, 5) numbers of nodes (for all levels of the tree), respectively.`
$\mathcal{L}_{\text{log\_probs\_draft}} \leftarrow [\,], \mathcal{L}_{\text{flat\_node\_ids}} \leftarrow [\,], \mathcal{L}_{\text{parent\_ids}} \leftarrow [\,], \mathcal{L}_{\text{draft\_tokens}} \leftarrow [\,], \mathcal{L}_{\text{num\_nodes}} \leftarrow [\,]$.

4: `// Initialize a draft tree.`
$\mathcal{T} \leftarrow (\mathcal{L}_{\text{log\_probs\_draft}}, \mathcal{L}_{\text{flat\_node\_ids}}, \mathcal{L}_{\text{parent\_ids}}, \mathcal{L}_{\text{draft\_tokens}})$.

5: `// Set an empty attention mask, and position ids; inclusive for start and exclusive for end.`
$\mathbf{M} \leftarrow \emptyset, \mathbf{id}_{\text{position}} \leftarrow \texttt{Arange}(\texttt{start} = 0, \texttt{end} = L_{\text{input}})$.

6: `// Set the counter to check the number of nodes in the tree.`
$N_{\text{tree\_prev}} \leftarrow 0, N_{\text{tree\_curr}} \leftarrow 0$.

7: `// Set the number of nodes at the current level of the tree.`
$N_{\text{nodes}} \leftarrow 1, \mathcal{L}_{\text{num\_nodes}}.\texttt{append}(N_{\text{nodes}})$.

8: **for** $l_{\text{draft}} = 0$ **to** $L_{\text{draft}} - 1$ **do**

9: `// Apply` $\mathbf{M}$ `to the right below corner of attention weights.`
`// The draft log probability` $\mathbf{\Phi}_{\text{draft}}$ `is a` $\texttt{GetLength}(\mathbf{x}_{\text{input}}) \times N_{\text{vocab}}$ `tensor.`
`//` N_{vocab} `is the vocabulary size.`
$\mathbf{\Phi}_{\text{draft}}, \mathbf{C}_{\text{draft}} \leftarrow \texttt{DraftModelForwardPass}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{id}_{\text{position}}, \mathbf{M})$.

10: `// Sample` $b_{l_{\text{draft}}}$ `nodes without replacement, independently for` N_{nodes} `nodes.`
`// NOTE: Outputs are sorted w.r.t. the value of perturbed log probabilities and flattened.`
$\mathbf{x}_{\text{draft}}, \mathbf{id}_{\text{parent}} \leftarrow \texttt{SampleWithGumbelTopK}(\mathbf{\Phi}_{\text{draft}}[-N_{\text{nodes}} :, :], b_{l_{\text{draft}}})$.

11: `// Update the input sequence of tokens.`
$\mathbf{x}_{\text{input}} \leftarrow \texttt{Concat}([\mathbf{x}_{\text{input}}, \mathbf{x}_{\text{draft}}])$.

12: `// Get the number of newly added nodes.`
$N_{\text{nodes}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{draft}})$.

13: `// Build attention mask reflecting tree topology.`
$\mathbf{M} \leftarrow \texttt{BuildAttentionMask}(\mathbf{M}, \mathbf{id}_{\text{parent}}, N_{\text{nodes}}, N_{\text{tree\_prev}}, N_{\text{tree\_curr}})$.

14: `// Update counters.`
$N_{\text{tree\_prev}} \leftarrow N_{\text{tree\_curr}}, N_{\text{tree\_curr}} \leftarrow N_{\text{tree\_curr}} + N_{\text{nodes}}$.

15: `// Update position IDs.`
$\mathbf{id}_{\text{position}} \leftarrow \texttt{Concat}([\mathbf{id}_{\text{position}}, (L_{\text{input}} + l_{\text{draft}}) \times \mathbf{1}_{N_{\text{nodes}}}])$.

16: `// Get node IDs considering the flattened draft tree.`
`// This is used to update KV caches.`
$\mathbf{id}_{\text{flat\_node}} \leftarrow \texttt{Arange}(\texttt{start} = L_{\text{input}} + N_{\text{tree\_prev}}, \texttt{end} = L_{\text{input}} + N_{\text{tree\_curr}})$.

17: `// Update the lists of the tree.`
$\mathcal{L}_{\text{log\_probs\_draft}}.\texttt{append}(\mathbf{\Phi}_{\text{draft}})$, $\mathcal{L}_{\text{flat\_node\_ids}}.\texttt{append}(\mathbf{id}_{\text{flat\_node}})$, $\mathcal{L}_{\text{parent\_ids}}.\texttt{append}(\mathbf{id}_{\text{parent}})$, $\mathcal{L}_{\text{draft\_tokens}}.\texttt{append}(\mathbf{x}_{\text{draft}})$, $\mathcal{L}_{\text{num\_nodes}}.\texttt{append}(N_{\text{nodes}})$.

18: **end for**

19: **Output:** $\mathcal{T}, \mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{M}, \mathbf{id}_{\text{position}}, \mathcal{L}_{\text{num\_nodes}}$.

Algorithm 4 `SampleWithGumbelTopK`$(\mathbf{\Phi}, K)$

Input: a $N_{\text{nodes}} \times N_{\text{vocab}}$ log probabilities $\mathbf{\Phi}$, the number K of desired samples without replacement.

// Sample a matrix where elements are i.i.d. standard Gumbel random variables.

$\mathbf{G} \leftarrow [g_{ij}], g_{ij} \leftarrow \texttt{SampleStandardGumbel}(), i = 0, ..., N_{\text{nodes}} - 1, j = 0, ..., N_{\text{vocab}} - 1.$

// Perturb log probabilities with Gumbel random variables.

$\tilde{\mathbf{\Phi}} \leftarrow \mathbf{\Phi} + \mathbf{G}.$

// Get top-K elements corresponding to the K largest perturb log probabilities.

// Outputs are sorted (in each row) w.r.t. the values of perturbed log probabilities and flattened.

$\mathbf{x} \leftarrow \text{argtop}^{(K)}(\tilde{\mathbf{\Phi}}, \texttt{dim} = -1).\texttt{flatten}().$

// Set parent ids.

$\mathbf{id}_{\text{parent}} \leftarrow \texttt{Concat}([0 \cdot \mathbf{1}_K, 1 \cdot \mathbf{1}_K, ..., (N_{\text{nodes}} - 1) \cdot \mathbf{1}_K]).$

// When probability filtering methods(e.g., top-p, top-k) were applied, filter some elements of $\mathbf{x}$ and $\mathbf{id}_{\text{parent}}$ if corresponding log probability is equal to $-\infty$.

Output: $\mathbf{x}, \mathbf{id}_{\text{parent}}.$

Algorithm 5 `BuildAttentionMask`$(\mathbf{M}, \mathbf{id}_{\text{parent}}, N_{\text{nodes}}, N_{\text{tree\_prev}}, N_{\text{tree\_curr}})$

Input: previous attention mask $\mathbf{M}$, parent node ids $\mathbf{id}_{\text{parent}}$ for newly added nodes, the number N_{nodes} of nodes newly added to the tree, the total number $N_{\text{tree\_prev}}$ of nodes in the previous-iteration tree, the total number $N_{\text{tree\_curr}}$ of nodes in the current-iteration tree.

if $\mathbf{M} == \emptyset$ **then**

  // If the attention mask is empty, we initialize with zeros.

  $\mathbf{M} \leftarrow \mathbf{0}_{N_{\text{nodes}} \times N_{\text{nodes}}}.$

else

  // If the attention mask exists, we zero-pad.

  $\mathbf{M} \leftarrow \texttt{ZeroPadding}(\mathbf{M}, \texttt{right} = N_{\text{nodes}}, \texttt{bottom} = N_{\text{nodes}}).$

  for $i = 0$ **to** $N_{\text{nodes}} - 1$ **do**

    // Copy the row about paraent nodes to the row about child nodes.

    $\mathbf{M}[N_{\text{tree\_curr}} + i, :] \leftarrow \mathbf{M}[N_{\text{tree\_prev}} + \mathbf{id}_{\text{parent}}[i], :].$

  end for

end if

// Set diagonal elements equal to 1.

$\mathbf{M} \leftarrow \mathbf{M}.\texttt{fill\_diagonal}(1)$

Output: the new attention mask $\mathbf{M}$.

Algorithm 6 $\texttt{RecursiveRejectionSampling}(\mathcal{T}, \mathcal{L}_{\text{log\_probs\_target}})$

Input: the draft tree $\mathcal{T}$, the list $\mathcal{L}_{\text{log\_probs\_target}}$ of target log probabilities
`// Get lists from the draft tree.`
$\mathcal{L}_{\text{log\_probs\_draft}}, \mathcal{L}_{\text{flat\_node\_ids}}, \mathcal{L}_{\text{parent\_ids}}, \mathcal{L}_{\text{draft\_tokens}} \leftarrow \mathcal{T}$
`// Set the current node id.`
$i_{node} \leftarrow 0$
`// Initialize the lists to store accepted draft tokens and flattened node ids (for KV cache update).`
$\mathcal{L}_{\text{accepted\_draft\_tokens}} \leftarrow [\,], \mathcal{L}_{\text{accepted\_flat\_node\_ids}} \leftarrow [\,]$.
for $l_{\text{draft}} = 0$ **to** $L_{\text{draft}} - 1$ **do**
  `// Get log probabilities at the current node.`
  `// Both are 1` $\times$ N_{vocab} `tensors, where` N_{vocab} `is the vocabulary size.`
  $\mathbf{\Phi}_{\text{draft}} \leftarrow \mathcal{L}_{\text{log\_probs\_draft}}[l_{\text{draft}}][i_{node} : (i_{node} + 1), :]$, $\mathbf{\Phi}_{\text{target}} \leftarrow \mathcal{L}_{\text{log\_probs\_target}}[l_{\text{draft}}][i_{node} : (i_{node} + 1), :]$
  `// Get draft tokens, flattened node IDs, parent IDs at the current level.`
  $\mathbf{x}_{\text{draft}} \leftarrow \mathcal{L}_{\text{draft\_tokens}}[l_{\text{draft}}]$, $\mathbf{id}_{\text{flat\_node}} \leftarrow \mathcal{L}_{\text{flat\_node\_ids}}[l_{\text{draft}}]$, $\mathbf{id}_{\text{parent}} \leftarrow \mathcal{L}_{\text{parent\_ids}}[l_{\text{draft}}]$
  `// Initialize an acceptance indicator as False.`
  accept $\leftarrow$ False
  for i **in** $\mathbf{id}_{\text{parent}}$ **do**
    if $i \neq i_{node}$ **then**
      continue
    end if
    `// Get the current draft token.`
    $x_d \leftarrow \mathbf{x}_{\text{draft}}[i]$.
    `// Sample a uniform random variable.`
    $U \sim \text{Uniform}[0, 1]$.
    if $U < \min\{1, \exp(\mathbf{\Phi}_{\text{target}}[0, x_d] - \mathbf{\Phi}_{\text{draft}}[0, x_d])\}$ **then**
      `// Set the indicator as True is the token is accepted.`
      accept $\leftarrow$ True.
      `// Store the accepted token and corresponding flattened node ID.`
      $\mathcal{L}_{\text{accepted\_draft\_tokens}}.\texttt{append}(x_d)$, $\mathcal{L}_{\text{accepted\_draft\_tokens}}.\texttt{append}(\mathbf{id}_{\text{flat\_node}}[i])$.
      $i_{node} \leftarrow i$.
      break
    end if
    `// Get clamped target log probability.`
    $\mathbf{\Phi}_{\text{target}} \leftarrow \log((\exp(\mathbf{\Phi}_{\text{target}}) - \exp(\mathbf{\Phi}_{\text{draft}})).\texttt{clamp}(\texttt{min} = 0)\})$
    `// Normalize the clamped target log probability.`
    $\mathbf{\Phi}_{\text{target}} \leftarrow \mathbf{\Phi}_{\text{target}} - \texttt{LogSumExp}(\mathbf{\Phi}_{\text{target}})$
    `// Neglect draft log probability of already sampled token.`
    $\mathbf{\Phi}_{\text{draft}}[i] \leftarrow -\infty$
    `// Normalize the draft log probability.`
    $\mathbf{\Phi}_{\text{draft}} \leftarrow \mathbf{\Phi}_{\text{draft}} - \texttt{LogSumExp}(\mathbf{\Phi}_{\text{draft}})$
  end for
  if accept == False **then**
    break
  end if
end for
if accept **then**
  `// At the leaf node when all tokens are accepted, we use target log probability to draw a sample.`
  $\mathbf{\Phi}_{\text{target}} \leftarrow \mathcal{L}_{\text{log\_probs\_target}}[l_d][i_{node} : (i_{node} + 1), :]$
end if
$\mathbf{x}_{\text{last}} \sim \texttt{SampleWithGumbelTopK}(\mathbf{\Phi}_{\text{target}}, 1)$
$\mathbf{x}_{\text{accepted}} \leftarrow \texttt{Stack}(\mathcal{L}_{\text{accepted\_draft\_tokens}}), \mathbf{id}_{\text{accepted\_flat\_node}} \leftarrow \texttt{Stack}(\mathcal{L}_{\text{accepted\_draft\_tokens}})$.
Output: $\mathbf{x}_{\text{accepted}}, \mathbf{x}_{\text{last}}, \mathbf{id}_{\text{accepted\_flat\_node}}$

C.2 Recursive Speculative Decoding with Stochastic Beam Search (RSD-S)

We highlight the difference w.r.t. RSD-C.

Algorithm 7 Recursive Speculative Decoding with Stochastic Beam Search (RSD-S)

1: **Input:** The length L_{draft} of draft sequences (depth of the draft tree), a sequence $\mathbf{x}_{\text{input}}$ of input tokens, the beamwidth W, the maximum length L_{output} of the output sequence.
2: `// Get the length of the input sequence.`
$L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$.
3: `// Initialize empty KV caches for draft and target models.`
$\mathbf{C}_{\text{draft}} \leftarrow \emptyset$, $\mathbf{C}_{\text{target}} \leftarrow \emptyset$.
4: **while** $L_{\text{input}} < L_{\text{output}}$ **do**
5: `// (STEP 1) Create a draft tree by using the draft model.`
$\mathcal{T}, \mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{M}, \mathbf{id}_{\text{position}}, \mathcal{L}_{\text{num\_nodes}}$
$\leftarrow \texttt{CreateDraftTreeStochasticBeamSearch}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, W, L_{\text{draft}})$.
6: `// (STEP 2) Evaluate draft tokens by using the target model.`
`// - Apply` $\mathbf{M}$ `to the right below corner of attention weights.`
`// - The target log probability` $\mathbf{\Phi}_{\text{target}}$ `is a` $\texttt{GetLength}(\mathbf{x}_{\text{input}}) \times N_{\text{vocab}}$ `tensor.`
`// -` N_{vocab} `is the vocabulary size.`
$\mathbf{\Phi}_{\text{target}}, \mathbf{C}_{\text{target}} \leftarrow \texttt{TargetModelForwardPass}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{target}}, \mathbf{id}_{\text{position}}, \mathbf{M})$.
7: `// - Convert the log probability tensor into the list of log probabilities for each level of the tree.`
$\mathcal{L}_{\text{log\_probs\_target}} \leftarrow \texttt{SplitTensor}(\mathbf{\Phi}_{\text{target}}[-\texttt{Sum}(\mathcal{L}_{\text{num\_nodes}}) :, :], \mathcal{L}_{\text{num\_nodes}}, \texttt{dim} = 0)$
8: `// (STEP 3) Run Recursive Rejection Sampling for each level of the tree.`
$\mathbf{x}_{\text{accepted}}, \mathbf{x}_{\text{last}}, \mathbf{id}_{\text{accepted\_flat\_node}} \leftarrow \texttt{RecursiveRejectionSampling}(\mathcal{T}, \mathcal{L}_{\text{log\_probs\_target}})$
9: `// (STEP 4) Use KV caches that are accepted, and prepare for the next round.`
$\mathbf{C}_{\text{draft}}, \mathbf{C}_{\text{target}} \leftarrow \texttt{FilterKVCache}(\mathbf{C}_{\text{draft}}, \mathbf{C}_{\text{target}}, L_{\text{input}}, \mathbf{id}_{\text{accepted\_flat\_node}})$
10: $\mathbf{x}_{\text{input}} \leftarrow \texttt{Concat}([\mathbf{x}_{\text{input}}[: L_{\text{input}}], \mathbf{x}_{\text{accepted}}, \mathbf{x}_{\text{last}}])$
11: $L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$
12: **end while**
13: **Output:** a sequence $\mathbf{x}_{\text{input}}$ that includes both input tokens and generated output tokens.

Algorithm 8 $\texttt{CreateDraftTreeStochasticBeamSearch}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, W, L_{\text{draft}})$

1: **Input:** An input sequence $\mathbf{x}_{\text{input}}$, the draft KV cache $\mathbf{C}_{\text{draft}}$, the beamwidth W, the draft length L_{draft}

2: `// Get the length of the input sequence.`
$L_{\text{input}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{input}})$.

3: `// Initialize lists for 1) draft log probabilities, 2) flattened node IDs, 3) parent node ids (within each level of the draft tree), 4) draft tokens, 5) numbers of nodes (for all levels of the tree), respectively.`
$\mathcal{L}_{\text{log\_probs\_draft}} \leftarrow [\,], \mathcal{L}_{\text{flat\_node\_ids}} \leftarrow [\,], \mathcal{L}_{\text{parent\_ids}} \leftarrow [\,], \mathcal{L}_{\text{draft\_tokens}} \leftarrow [\,], \mathcal{L}_{\text{num\_nodes}} \leftarrow [\,]$.

4: `// Initialize a draft tree.`
$\mathcal{T} \leftarrow (\mathcal{L}_{\text{log\_probs\_draft}}, \mathcal{L}_{\text{flat\_node\_ids}}, \mathcal{L}_{\text{parent\_ids}}, \mathcal{L}_{\text{draft\_tokens}})$.

5: `// Set an empty attention mask, and position ids; inclusive for start and exclusive for end.`
$\mathbf{M} \leftarrow \emptyset, \mathbf{id}_{\text{position}} \leftarrow \texttt{Arange}(\texttt{start} = 0, \texttt{end} = L_{\text{input}})$.

6: `// Set the counter to check the number of nodes in the tree.`
$N_{\text{tree\_prev}} \leftarrow 0, N_{\text{tree\_curr}} \leftarrow 0$.

7: `// Set the number of nodes at the current level of the tree.`
$N_{\text{nodes}} \leftarrow 1, \mathcal{L}_{\text{num\_nodes}}.\texttt{append}(N_{\text{nodes}})$.

8: `// Set stochastic beam parameters: sum log probabilities` $\boldsymbol{\Sigma}$ `and truncated Gumbels` $\boldsymbol{\Gamma}$ `for each node in the current level of draft tree`
$\boldsymbol{\Sigma} \leftarrow \mathbf{0}_{N_{\text{nodes}} \times 1}, \boldsymbol{\Gamma} \leftarrow \mathbf{0}_{N_{\text{nodes}} \times 1}$.

9: **for** $l_{\text{draft}} = 0$ **to** $L_{\text{draft}} - 1$ **do**

10: `// Apply` $\mathbf{M}$ `to the right below corner of attention weights.`
`// The draft log probability` $\boldsymbol{\Phi}_{\text{draft}}$ `is a` $\texttt{GetLength}(\mathbf{x}_{\text{input}}) \times N_{\text{vocab}}$ `tensor.`
`//` N_{vocab} `is the vocabulary size.`
$\boldsymbol{\Phi}_{\text{draft}}, \mathbf{C}_{\text{draft}} \leftarrow \texttt{DraftModelForwardPass}(\mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{id}_{\text{position}}, \mathbf{M})$.

11: `// Sample` $b_{l_{\text{draft}}}$ `nodes without replacement, independently for` N_{nodes} `nodes.`
`// NOTE: Outputs are sorted w.r.t. the value of perturbed log probabilities and flattened.`
$\mathbf{x}_{\text{draft}}, \mathbf{id}_{\text{parent}}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma} \leftarrow \texttt{SampleWithStochasticBeam}(\boldsymbol{\Phi}_{\text{draft}}[-N_{\text{nodes}}:,:], \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, W)$.

12: `// Update the input sequence of tokens.`
$\mathbf{x}_{\text{input}} \leftarrow \texttt{Concat}([\mathbf{x}_{\text{input}}, \mathbf{x}_{\text{draft}}])$.

13: `// Get the number of newly added nodes.`
$N_{\text{nodes}} \leftarrow \texttt{GetLength}(\mathbf{x}_{\text{draft}})$.

14: `// Build attention mask reflecting tree topology.`
$\mathbf{M} \leftarrow \texttt{BuildAttentionMask}(\mathbf{M}, \mathbf{id}_{\text{parent}}, N_{\text{nodes}}, N_{\text{tree\_prev}}, N_{\text{tree\_curr}})$.

15: `// Update counters.`
$N_{\text{tree\_prev}} \leftarrow N_{\text{tree\_curr}}, N_{\text{tree\_curr}} \leftarrow N_{\text{tree\_curr}} + N_{\text{nodes}}$.

16: `// Update position IDs.`
$\mathbf{id}_{\text{position}} \leftarrow \texttt{Concat}([\mathbf{id}_{\text{position}}, (L_{\text{input}} + l_{\text{draft}}) \times \mathbf{1}_{N_{\text{nodes}}}])$.

17: `// Get node IDs considering the flattened draft tree.`
`// This is used to update KV caches.`
$\mathbf{id}_{\text{flat\_node}} \leftarrow \texttt{Arange}(\texttt{start} = L_{\text{input}} + N_{\text{tree\_prev}}, \texttt{end} = L_{\text{input}} + N_{\text{tree\_curr}})$.

18: `// Update the lists of the tree.`
$\mathcal{L}_{\text{log\_probs\_draft}}.\texttt{append}(\boldsymbol{\Phi}_{\text{draft}})$, $\mathcal{L}_{\text{flat\_node\_ids}}.\texttt{append}(\mathbf{id}_{\text{flat\_node}})$, $\mathcal{L}_{\text{parent\_ids}}.\texttt{append}(\mathbf{id}_{\text{parent}})$,
$\mathcal{L}_{\text{draft\_tokens}}.\texttt{append}(\mathbf{x}_{\text{draft}})$, $\mathcal{L}_{\text{num\_nodes}}.\texttt{append}(N_{\text{nodes}})$.

19: **end for**

20: **Output:** $\mathcal{T}, \mathbf{x}_{\text{input}}, \mathbf{C}_{\text{draft}}, \mathbf{M}, \mathbf{id}_{\text{position}}, \mathcal{L}_{\text{num\_nodes}}$.

Algorithm 9 SampleWithStochasticBeam($\mathbf{\Phi}, \mathbf{\Sigma}, \mathbf{\Gamma}, K$)

Input: a $N_{\text{nodes}} \times N_{\text{vocab}}$ log probabilities $\mathbf{\Phi}$, a $N_{\text{nodes}} \times 1$ sum log probabilities $\mathbf{\Sigma}$, a $N_{\text{nodes}} \times 1$ truncated Gumbels $\mathbf{\Gamma}$, the beamwidth K.

// Get sum log probs up to child nodes.
$\mathbf{\Phi} \leftarrow \mathbf{\Phi} + \mathbf{\Sigma} \mathbf{1}_{1 \times N_{\text{vocab}}}$.

// Sample a matrix where elements are i.i.d. standard Gumbel random variables.
$\mathbf{G} \leftarrow [g_{ij}], g_{ij} \leftarrow \texttt{SampleStandardGumbel}(), i = 0, ..., N_{\text{nodes}} - 1, j = 0, ..., N_{\text{vocab}} - 1.$

// Perturb sum log probabilities with Gumbel random variables.
$\tilde{\mathbf{\Phi}} \leftarrow \mathbf{\Phi} + \mathbf{G}$.

// Compute row-wise maximum value of perturbed sum log probabilities.
// The output size is $N_{\text{nodes}} \times 1$.
$\tilde{\mathbf{\Phi}}_{\text{max}} \leftarrow \tilde{\mathbf{\Phi}}.\texttt{max}(\texttt{dim} = -1, \texttt{keepdim} = \text{True})$.

// Get truncated Gumbels for all expansion.
// The output size is $N_{\text{nodes}} \times N_{\text{vocab}}$.
// NOTE: the numerical stable way of computing this quantity was described in the original Stochastic Beam Search paper.
$\tilde{\mathbf{\Gamma}} \leftarrow -\log(\exp(-\mathbf{\Gamma} \mathbf{1}_{1 \times N_{\text{vocab}}}) - \exp(-\tilde{\mathbf{\Phi}}_{\text{max}} \mathbf{1}_{1 \times N_{\text{vocab}}}) + \exp(-\tilde{\mathbf{\Phi}}))$

// Get top-K elements and the K largest truncated Gumbels.
// NOTE: we consider top-K elements for all elements in $\tilde{\mathbf{\Gamma}}$, so both parent node IDs and token IDs can be acquired. Make sure that both output IDs are sorted w.r.t. the corresponding values in $\tilde{\mathbf{\Gamma}}$.
$\mathbf{id}_{\text{parent}}, \mathbf{x}, \mathbf{\Gamma} \leftarrow \text{argtop-}K(\tilde{\mathbf{\Phi}})$.

// Get sum log probs for top-K elements.
$\mathbf{\Sigma} \leftarrow \mathbf{\Phi}[\mathbf{id}_{\text{parent}}, \mathbf{x}]$.

// When probability filtering methods (e.g., top-p, top-k) were applied, filter some elements of $\mathbf{x}, \mathbf{id}_{\text{parent}}, \mathbf{\Sigma}, \mathbf{\Gamma}$ if corresponding log probability is equal to $-\infty$.

Output: $\mathbf{x}, \mathbf{id}_{\text{parent}}, \mathbf{\Sigma}, \mathbf{\Gamma}$

D Experiments

D.1 Draft models

The following draft models are used:

- For Llama 2 target models, we use the 115M **Llama 2 drafter** and **Llama 2-Chat drafter** for **Llama 2** and **Llama 2-Chat** target models, respectively.
  - **Llama 2 drafter** uses smaller Llama archiecture (Touvron et al., 2023) and is pre-trained on the 600B-token dataset
  - **Llama 2-Chat drafter** is the model fine-tuned from Llama 2-drafter so that it can be aligned with Llama 2-Chat-7B via distillation.
- For OPT target models, we use **OPT** with **125M** and **350M** parameters for target OPT models.

D.2 Performance Metrics

In the experiments, we consider three metrics (except accuracy) for all target models.

- **Block efficiency** (Leviathan et al., 2023) is the average number of tokens generated per target model call. Within a single target call, auto-regressive decoding always generates a single token, while speculative decoding methods generates

$$\text{(Number of accepted tokens)} + 1.$$

  The block efficiency η is the average over all target calls.
- **Memory-Bound Speed Up (MBSU)** is the fictitious inference speed-up relative to auto-regressive decoding, where we assume each model's runtime is proportional to the model size. Specifically, let L denote the (maximum) length of draft sequences, which is the depth of the draft-token tree for tree-based speculative decoding methods, and r denote the relative speed of running the draft model to that of the target model. The walltime improvement (Leviathan et al., 2023; Zhou et al., 2023) is

$$\frac{\eta}{L \times r + 1}.$$

  MBSU considers a specific case where r is equal to $\text{(Size of the target model)}/\text{(Size of the draft model)}$, considering practical scenarios in memory-bound devices where loading model weights takes significant amount time, often proportional to their size.
- **Token rate** is the measure of average number of generated tokens per second while running on A100 GPUs. It shows different results from MBSU since running A100 GPUs is far from memory-bound scenarios.

D.3 Tree Structure

D.3.1 Experiment for various lengths of draft sequence

The following trees are used for draft sequence length L, where SD uses a single draft sequence with length L. For each L, we first set RSD-C with constant branching factors always equal to 2 and set the draft-tree sizes for SpecTr and RSD-S *always less than or equal to* the tree size of RSD-C. Then, we add RSD-C with the branching factor $\mathbf{b} := [n, 1, ..., 1]$ where n is properly set to have the draft-tree size equal to that of SpecTr and RSD-S. In *Figure 4*, we show the best results across all tree structures for each L and algorithm.

- $L = 2$:
  - SpecTr and RSD-S: $(K, L) \in \{(2, 2), (3, 2)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 2], [2, 1], [3, 1]\}$ for a vector $\mathbf{b}$ of branching factors.

- $L = 3$
  - SpecTr and RSD-S: $(K, L) \in \{(3, 3), (4, 3)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 2, 2], [3, 1, 1], [4, 1, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $L = 4$
  - SpecTr and RSD-S: $(K, L) \in \{(5, 4), (7, 4)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 2, 2, 2], [5, 1, 1, 1], [7, 1, 1, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $L = 5$
  - SpecTr and RSD-S: $(K, L) \in \{(6, 5), (12, 5)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 2, 2, 2, 2], [6, 1, 1, 1, 1], [12, 1, 1, 1, 1]\}$ for a vector $\mathbf{b}$ of branching factors.

D.3.2 Experiment for vairous target computational budget

The following trees are used for target computational budgets B, i.e., the number of tokens to process at the target model, where B becomes the draft length of SD. In *Figure 5*, we show the best results across all tree structures for each B and algorithm.

- $B = 6$
  - SpecTr and RSD-S: $(K, L) \in \{(2, 3), (3, 2)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 1, 1], [2, 2], [3, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $B = 10$
  - SpecTr and RSD-S: $(K, L) \in \{(2, 5), (5, 2)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 1, 1, 1, 1], [2, 2, 1], [5, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $B = 14$
  - SpecTr and RSD-S: $(K, L) \in \{(2, 7), (7, 2)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 1, 1, 1, 1, 1, 1], [2, 2, 2], [7, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $B = 21$
  - SpecTr and RSD-S: $(K, L) \in \{(3, 7), (7, 3)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[3, 1, 1, 1, 1, 1, 1], [3, 2, 2], [7, 1, 1]\}$ for a vector $\mathbf{b}$ of branching factors.
- $B = 30$
  - SpecTr and RSD-S: $(K, L) \in \{(5, 6), (6, 5)\}$, where K becomes the number of independent draft sequences for SpecTr and the beamwidth for RSD-S
  - RSD-C: $\mathbf{b} \in \{[2, 2, 2, 2], [5, 1, 1, 1, 1, 1], [6, 1, 1, 1, 1]\}$ for a vector $\mathbf{b}$ of branching factors.

D.4 Experiment results with plots

D.4.1 Block efficiency, MBSU, token rate and accuracy for various lengths of draft sequence

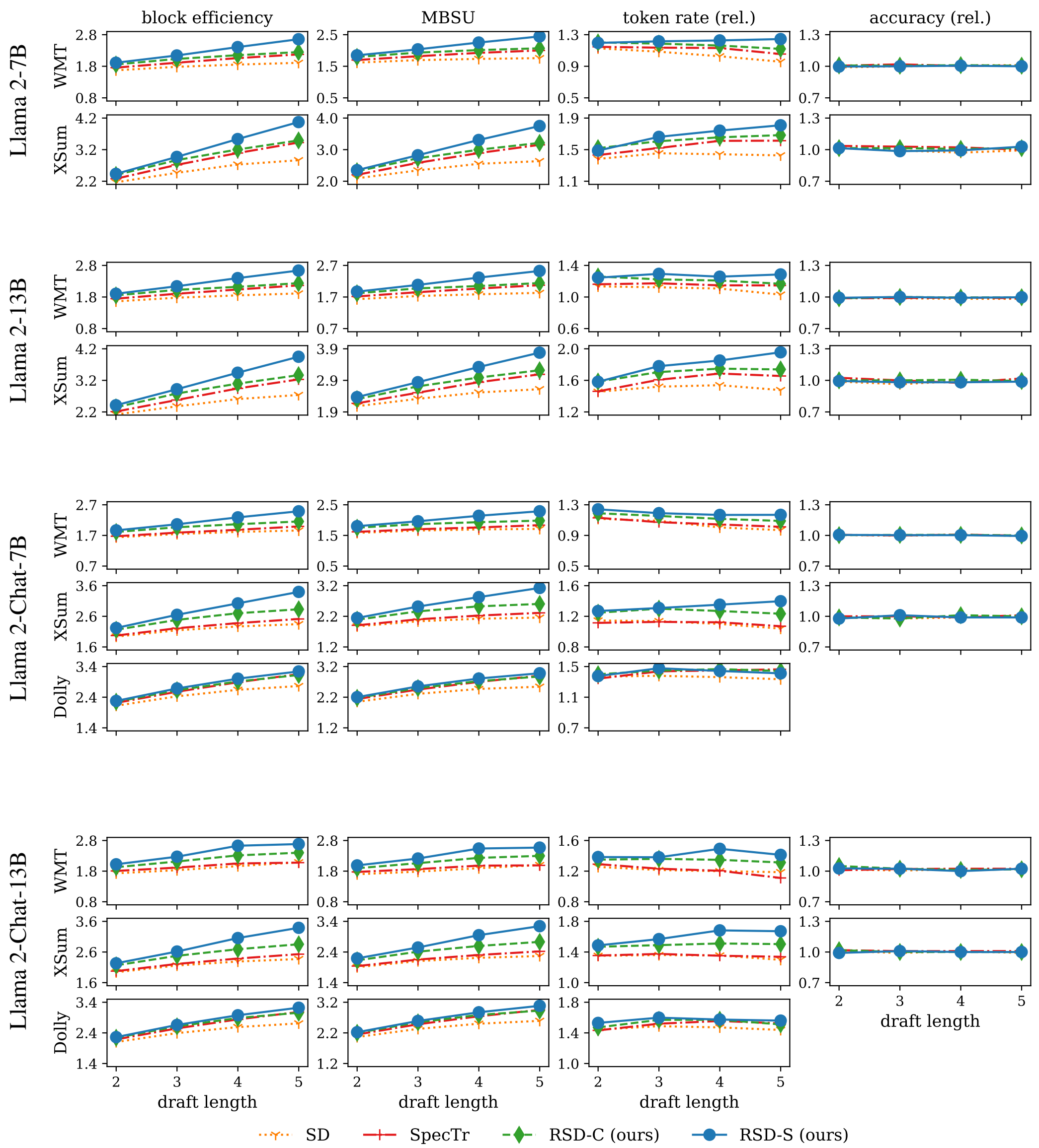


Figure 6: Block efficiency, MBSU, token rate and accuracy for varying lengths of draft sequence are given for multiple target models: Llama 2-7B, Llama 2-13B, Llama 2-Chat-7B, Llama 2-Chat-13B. Chat models use the same draft model, while the other models use the same draft model different from the one for chat models. All results are normalized w.r.t. the values of AR decoding.

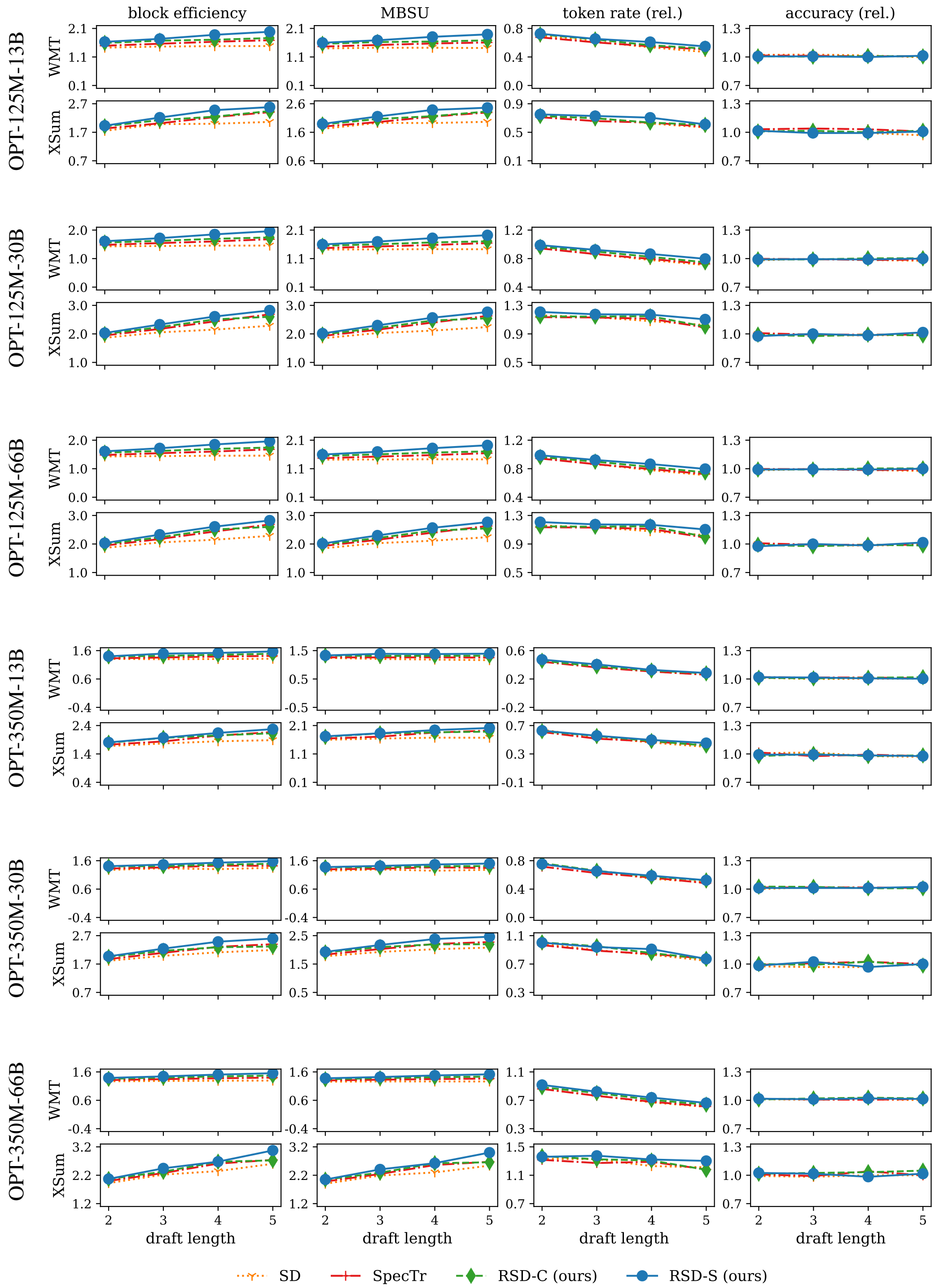


Figure 7: Block efficiency, MBSU, token rate and accuracy for varying lengths of draft sequence are given for multiple pairs of draft and target models: the size of draft model is in {125M, 350M}, and the size of target model is in {13B, 30B, 66B}. All results are normalized w.r.t. the values of AR decoding.

D.4.2 BLOCK EFFICIENCY, MBSU, TOKEN RATE AND ACCURACY FOR VARIOUS TARGET COMPUTATIONAL BUDGET

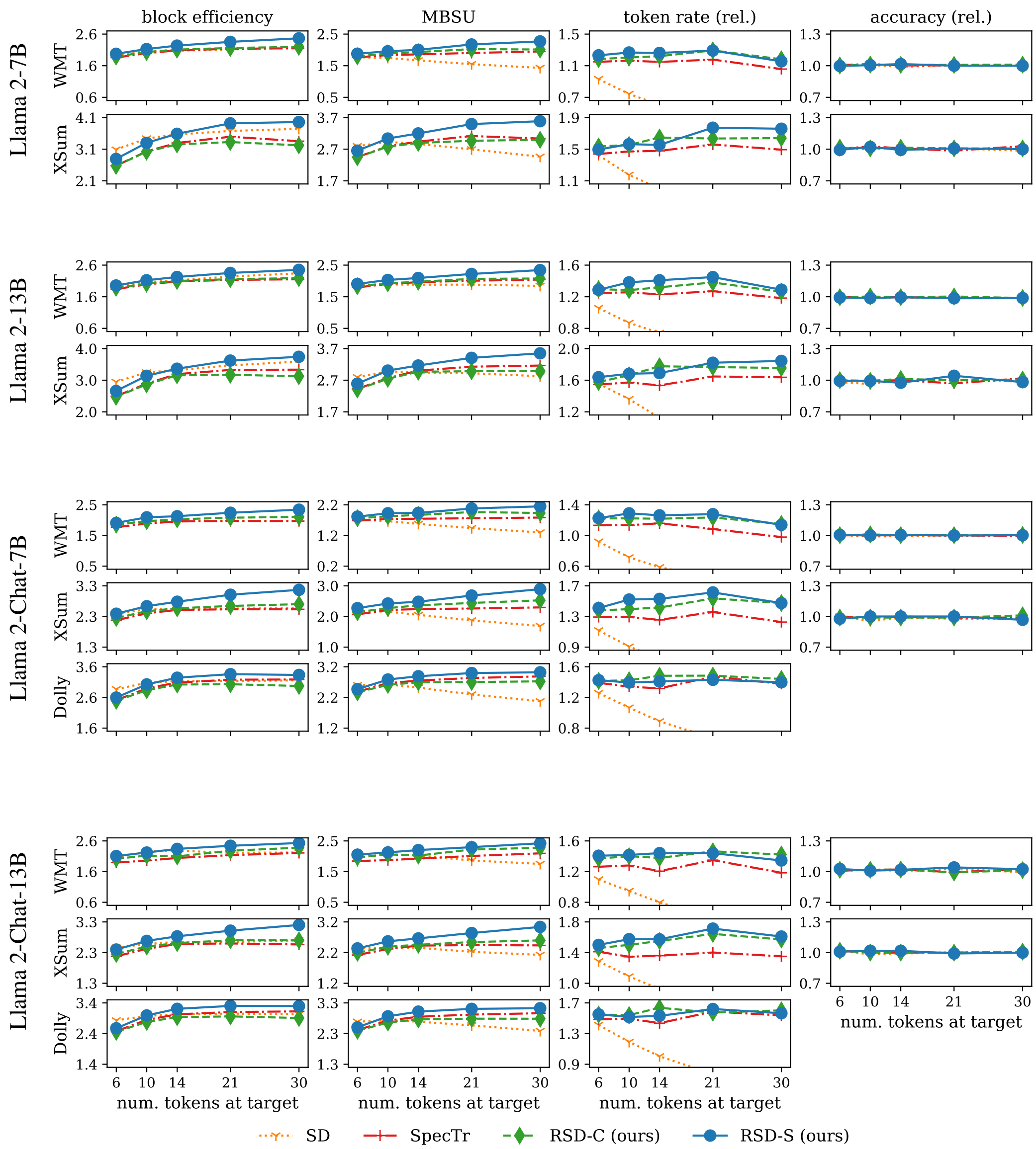


Figure 8: Block efficiency, MBSU, token rate and accuracy for varying numbers of tokens processed at the target model are given for multiple target models: Llama 2-7B, Llama 2-13B, Llama 2-Chat-7B, Llama 2-Chat-13B. Chat models use the same draft model, while the other models use the same draft model different from the one for chat models. All results are normalized w.r.t. the values of AR decoding.

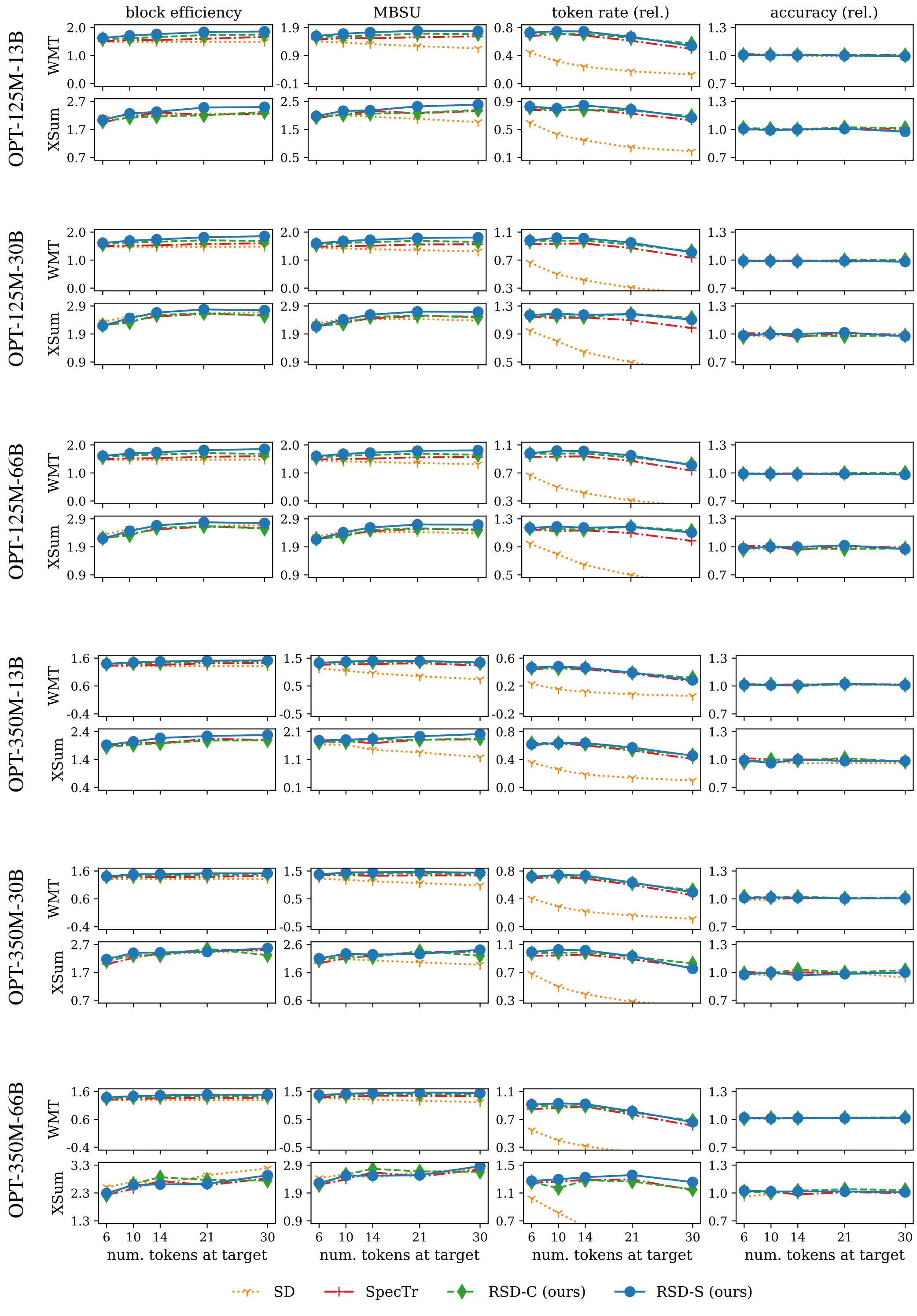


Figure 9: Block efficiency, MBSU, token rate and accuracy for varying numbers of tokens processed at the target model are given for multiple pairs of draft and target models: the size of draft model is in {125M, 350M}, and the size of target model is in {13B, 30B, 66B}. All results are normalized w.r.t. the values of AR decoding.

D.5 Experient results with tables

For readers curious about raw numbers, we remain all the numbers used for plots as tables in this section.

D.5.1 Block efficiency, MBSU, token rate and accuracy for varying lengths of draft sequence

- Llama 2-7B (with 115M drafter)
  - XSum (*Table* 1), WMT (*Table* 2)
- Llama 2-13B (with 115M drafter)
  - XSum (*Table* 3), WMT (*Table* 4)
- Llama 2-70B (with 115M drafter)
  - XSum (*Table* 5), WMT (*Table* 6)
- Llama 2-Chat-7B (with 115M drafter)
  - XSum (*Table* 7), WMT (*Table* 8), Dolly (*Table* 9)
- Llama 2-Chat-13B (with 115M drafter)
  - XSum (*Table* 10), WMT (*Table* 11), Dolly (*Table* 12)
- Llama 2-Chat-70B (with 115M drafter)
  - XSum (*Table* 13), WMT (*Table* 14), Dolly (*Table* 15)
- OPT-13B (with OPT-125M drafter)
  - XSum (*Table* 16), WMT (*Table* 17)
- OPT-30B (with OPT-125M drafter)
  - XSum (*Table* 18), WMT (*Table* 19)
- OPT-66B (with OPT-125M drafter)
  - XSum (*Table* 20), WMT (*Table* 21)
- OPT-13B (with OPT-350M drafter)
  - XSum (*Table* 22), WMT (*Table* 23)
- OPT-30B (with OPT-350M drafter)
  - XSum (*Table* 24), WMT (*Table* 25)
- OPT-66B (with OPT-350M drafter)
  - XSum (*Table* 26), WMT (*Table* 27)

D.5.2 Block efficiency, MBSU, token rate and accuracy for varying numbers of tokens processed at the target model

- Llama 2-7B (with 115M drafter)
  - XSum (*Table* 28), WMT (*Table* 29)
- Llama 2-13B (with 115M drafter)
  - XSum (*Table* 30), WMT (*Table* 31)
- Llama 2-70B (with 115M drafter)
  - XSum (*Table* 32), WMT (*Table* 33)
- Llama 2-Chat-7B (with 115M drafter)
  - XSum (*Table* 34), WMT (*Table* 35), Dolly (*Table* 36)
- Llama 2-Chat-13B (with 115M drafter)
  - XSum (*Table* 37), WMT (*Table* 38), Dolly (*Table* 39)
- Llama 2-Chat-70B (with 115M drafter)
  - XSum (*Table* 40), WMT (*Table* 41), Dolly (*Table* 42)
- OPT-13B (with OPT-125M drafter)
  - XSum (*Table* 43), WMT (*Table* 44)
- OPT-30B (with OPT-125M drafter)
  - XSum (*Table* 45), WMT (*Table* 46)
- OPT-66B (with OPT-125M drafter)
  - XSum (*Table* 47), WMT (*Table* 48)
- OPT-13B (with OPT-350M drafter)
  - XSum (*Table* 49), WMT (*Table* 50)
- OPT-30B (with OPT-350M drafter)
  - XSum (*Table* 51), WMT (*Table* 52)
- OPT-66B (with OPT-350M drafter)
  - XSum (*Table* 53), WMT (*Table* 54)

Table 1: We summarize experiment results with Llama 2-7B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-7B	XSum	0	AR	-	1.000	1.000	37.269	0.141
		2	SD	2	2.166	2.093	51.515	0.143
			SpecTr	2×2	2.218	2.143	52.950	0.146
				3×2	2.279	2.202	53.346	0.139
			RSD-C	2−1	2.267	2.191	53.980	0.142
				2−2	2.398	2.317	**56.609**	0.143
				3−1	2.291	2.214	53.930	0.140
			RSD-S	2×2	2.367	2.288	54.586	0.143
				3×2	**2.432**	**2.350**	55.465	0.140
		3	SD	3	2.465	2.343	54.195	0.140
			SpecTr	3×3	2.644	2.513	55.273	0.140
				4×3	2.718	2.583	56.688	0.145
			RSD-C	2−2−2	2.868	2.726	59.879	0.141
				3−1−1	2.641	2.511	55.384	0.143
				4−1−1	2.688	2.555	57.518	0.140
			RSD-S	3×3	2.927	2.782	58.843	0.139
				4×3	**2.970**	**2.823**	**61.937**	0.136
		4	SD	4	2.728	2.551	53.731	0.137
			SpecTr	5×4	2.974	2.781	56.002	0.144
				7×4	3.093	2.892	60.053	0.139
			RSD-C	2−2−2−2	3.205	2.997	61.723	0.142
				5−1−1−1	2.898	2.710	56.343	0.141
				7−1−1−1	2.974	2.781	58.423	0.137
			RSD-S	5×4	3.427	3.205	**64.887**	0.140
				7×4	**3.535**	**3.306**	64.456	0.140
		5	SD	5	2.865	2.636	53.199	0.140
			SpecTr	6×5	3.209	2.953	55.424	0.141
				12×5	3.425	3.152	60.133	0.141
			RSD-C	2−2−2−2−2	3.492	3.213	62.753	0.143
				6−1−1−1−1	3.133	2.883	55.796	0.142
				12−1−1−1−1	3.249	2.990	58.352	0.141
			RSD-S	6×5	3.811	3.507	65.160	0.141
				12×5	**4.073**	**3.748**	**67.409**	0.145

Table 2: We summarize experiment results with Llama 2-7B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-7B	WMT	0	AR	-	1.000	1.000	37.631	0.374
		2	SD	2	1.673	1.617	42.447	0.370
			SpecTr	2×2	1.727	1.669	42.013	0.370
				3×2	1.757	1.698	43.128	0.376
			RSD-C	2−1	1.768	1.708	43.044	0.377
				2−2	1.858	1.796	**45.245**	0.372
				3−1	1.819	1.758	44.482	0.375
			RSD-S	2×2	1.824	1.763	43.536	0.370
				3×2	**1.912**	**1.847**	45.018	0.373
		3	SD	3	1.783	1.695	40.816	0.374
			SpecTr	3×3	1.890	1.796	42.746	0.381
				4×3	1.913	1.819	41.990	0.379
			RSD-C	2−2−2	2.033	1.933	44.669	0.372
				3−1−1	1.940	1.844	42.981	0.370
				4−1−1	1.981	1.883	43.791	0.376
			RSD-S	3×3	2.064	1.962	43.684	0.372
				4×3	**2.143**	**2.037**	**45.766**	0.374
		4	SD	4	1.854	1.734	38.651	0.377
			SpecTr	5×4	2.023	1.892	41.134	0.375
				7×4	2.059	1.925	42.573	0.373
			RSD-C	2−2−2−2	2.152	2.013	43.755	0.378
				5−1−1−1	2.083	1.948	43.142	0.375
				7−1−1−1	2.130	1.992	42.567	0.375
			RSD-S	5×4	2.311	2.161	44.367	0.375
				7×4	**2.408**	**2.252**	**46.197**	0.376
		5	SD	5	1.910	1.758	36.041	0.378
			SpecTr	6×5	2.120	1.951	38.841	0.373
				12×5	2.176	2.002	39.702	0.376
			RSD-C	2−2−2−2−2	2.234	2.056	42.161	0.375
				6−1−1−1−1	2.171	1.998	40.331	0.372
				12−1−1−1−1	2.249	2.070	41.585	0.376
			RSD-S	6×5	2.467	2.270	43.898	0.370
				12×5	**2.657**	**2.445**	**46.843**	0.374

Table 3: We summarize experiment results with Llama 2-13B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-13B	XSum	0	AR	-	1.000	1.000	28.141	0.166
		2	SD	2	2.120	2.082	40.958	0.164
			SpecTr	2×2	2.172	2.133	40.870	0.170
				3×2	2.212	2.173	41.116	0.165
			RSD-C	2−1	2.224	2.185	41.866	0.165
				2−2	2.347	2.305	**44.593**	0.166
				3−1	2.269	2.229	42.981	0.158
			RSD-S	2×2	2.311	2.271	43.533	0.165
				3×2	**2.412**	**2.370**	44.529	0.162
		3	SD	3	2.377	2.315	42.777	0.160
			SpecTr	3×3	2.559	2.492	45.252	0.166
				4×3	2.578	2.510	44.703	0.164
			RSD-C	2−2−2	2.784	2.711	47.985	0.166
				3−1−1	2.560	2.493	44.855	0.164
				4−1−1	2.593	2.525	44.639	0.161
			RSD-S	3×3	2.832	2.758	48.388	0.162
				4×3	**2.919**	**2.842**	**50.092**	0.163
		4	SD	4	2.608	2.517	43.309	0.165
			SpecTr	5×4	2.880	2.780	45.940	0.161
				7×4	2.944	2.842	47.456	0.162
			RSD-C	2−2−2−2	3.096	2.989	49.203	0.167
				5−1−1−1	2.813	2.715	44.641	0.165
				7−1−1−1	2.864	2.764	45.288	0.162
			RSD-S	5×4	3.347	3.231	50.517	0.163
				7×4	**3.442**	**3.322**	**52.105**	0.157
		5	SD	5	2.738	2.621	41.562	0.165
			SpecTr	6×5	3.108	2.974	46.120	0.169
				12×5	3.230	3.091	46.587	0.166
			RSD-C	2−2−2−2−2	3.365	3.220	48.923	0.165
				6−1−1−1−1	3.014	2.885	43.751	0.163
				12−1−1−1−1	3.069	2.937	44.262	0.164
			RSD-S	6×5	3.648	3.492	50.782	0.162
				12×5	**3.948**	**3.778**	**55.044**	0.164

Table 4: We summarize experiment results with Llama 2-13B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-13B	WMT	0	AR	-	1.000	1.000	30.467	0.413
		2	SD	2	1.662	1.632	34.571	0.410
			SpecTr	2×2	1.717	1.686	35.383	0.405
				3×2	1.748	1.717	35.124	0.408
			RSD-C	2−1	1.760	1.729	36.200	0.405
				2−2	1.852	1.819	**38.384**	0.407
				3−1	1.815	1.783	37.576	0.408
			RSD-S	2×2	1.810	1.778	35.906	0.404
				3×2	**1.903**	**1.869**	37.934	0.410
		3	SD	3	1.778	1.731	34.213	0.408
			SpecTr	3×3	1.876	1.827	35.754	0.407
				4×3	1.903	1.853	35.127	0.409
			RSD-C	2−2−2	2.027	1.974	36.916	0.407
				3−1−1	1.929	1.878	37.279	0.413
				4−1−1	1.965	1.914	35.558	0.408
			RSD-S	3×3	2.059	2.005	36.511	0.406
				4×3	**2.141**	**2.084**	**39.415**	0.413
		4	SD	4	1.852	1.787	33.728	0.406
			SpecTr	5×4	2.004	1.935	34.950	0.409
				7×4	2.038	1.968	34.973	0.411
			RSD-C	2−2−2−2	2.122	2.048	36.819	0.407
				5−1−1−1	2.080	2.008	35.434	0.411
				7−1−1−1	2.116	2.042	35.526	0.406
			RSD-S	5×4	2.304	2.224	37.842	0.408
				7×4	**2.399**	**2.315**	**38.315**	0.410
		5	SD	5	1.913	1.831	31.396	0.406
			SpecTr	6×5	2.101	2.011	34.184	0.408
				12×5	2.168	2.074	34.936	0.408
			RSD-C	2−2−2−2−2	2.198	2.103	34.472	0.412
				6−1−1−1−1	2.163	2.070	34.502	0.408
				12−1−1−1−1	2.238	2.142	35.575	0.410
			RSD-S	6×5	2.448	2.343	36.278	0.408
				12×5	**2.638**	**2.525**	**39.182**	0.412

Table 5: We summarize experiment results with Llama 2-70B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-70B	XSum	0	AR	-	1.000	1.000	9.079	0.194
		2	SD	2	2.103	2.096	15.054	0.188
			SpecTr	2×2	2.164	2.157	15.171	0.189
				3×2	2.204	2.197	15.346	0.191
			RSD-C	2−1	2.189	2.181	15.276	0.187
				2−2	2.322	2.314	16.033	0.189
				3−1	2.239	2.231	15.480	0.197
			RSD-S	2×2	2.288	2.280	15.719	0.189
				3×2	**2.376**	**2.368**	**16.284**	0.193
		3	SD	3	2.365	2.353	15.992	0.193
			SpecTr	3×3	2.528	2.515	16.533	0.195
				4×3	2.554	2.541	16.586	0.193
			RSD-C	2−2−2	2.757	2.743	17.790	0.188
				3−1−1	2.551	2.538	16.837	0.189
				4−1−1	2.543	2.531	16.617	0.196
			RSD-S	3×3	2.765	2.751	17.689	0.192
				4×3	**2.862**	**2.848**	**18.163**	0.186
		4	SD	4	2.584	2.566	16.656	0.196
			SpecTr	5×4	2.844	2.825	17.159	0.192
				7×4	2.883	2.863	17.052	0.192
			RSD-C	2−2−2−2	3.028	3.008	17.885	0.191
				5−1−1−1	2.749	2.730	16.658	0.193
				7−1−1−1	2.780	2.762	16.568	0.190
			RSD-S	5×4	3.242	3.220	18.965	0.196
				7×4	**3.361**	**3.338**	**19.248**	0.191
		5	SD	5	2.680	2.658	16.634	0.194
			SpecTr	6×5	3.103	3.077	17.603	0.192
				12×5	3.206	3.179	17.344	0.194
			RSD-C	2−2−2−2−2	3.295	3.268	17.675	0.191
				6−1−1−1−1	2.935	2.910	16.809	0.193
				12−1−1−1−1	3.004	2.978	16.371	0.193
			RSD-S	6×5	3.556	3.526	19.484	0.192
				12×5	**3.851**	**3.819**	**19.808**	0.194

Table 6: We summarize experiment results with Llama 2-70B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-70B	WMT	0	AR	-	1.000	1.000	9.706	0.439
		2	SD	2	1.661	1.655	13.331	0.440
			SpecTr	2×2	1.732	1.726	13.742	0.445
				3×2	1.756	1.750	13.710	0.445
			RSD-C	2−1	1.770	1.764	13.992	0.436
				2−2	1.853	1.847	14.512	0.443
				3−1	1.819	1.813	14.245	0.440
			RSD-S	2×2	1.819	1.813	14.211	0.438
				3×2	**1.907**	**1.900**	**14.727**	0.439
		3	SD	3	1.778	1.769	13.620	0.442
			SpecTr	3×3	1.880	1.870	13.906	0.440
				4×3	1.909	1.899	13.959	0.437
			RSD-C	2−2−2	2.021	2.010	14.875	0.438
				3−1−1	1.940	1.930	14.474	0.441
				4−1−1	1.968	1.958	14.483	0.439
			RSD-S	3×3	2.068	2.057	15.132	0.440
				4×3	**2.140**	**2.129**	**15.591**	0.437
		4	SD	4	1.866	1.854	13.698	0.437
			SpecTr	5×4	2.016	2.003	13.865	0.440
				7×4	2.048	2.034	13.833	0.441
			RSD-C	2−2−2−2	2.130	2.116	14.338	0.440
				5−1−1−1	2.088	2.074	14.311	0.440
				7−1−1−1	2.132	2.117	14.406	0.441
			RSD-S	5×4	2.309	2.293	15.550	0.434
				7×4	**2.408**	**2.392**	**15.962**	0.437
		5	SD	5	1.917	1.901	13.430	0.437
			SpecTr	6×5	2.113	2.095	13.737	0.443
				12×5	2.177	2.159	13.619	0.442
			RSD-C	2−2−2−2−2	2.232	2.214	13.988	0.443
				6−1−1−1−1	2.173	2.154	14.095	0.440
				12−1−1−1−1	2.246	2.227	14.022	0.440
			RSD-S	6×5	2.451	2.430	15.575	0.439
				12×5	**2.650**	**2.628**	**15.941**	0.439

Table 7: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	XSum	0	AR	-	1.000	1.000	41.651	0.092
		2	SD	2	1.938	1.873	47.708	0.091
			SpecTr	2×2	1.961	1.896	46.422	0.092
				3×2	1.972	1.906	45.886	0.092
			RSD-C	2−1	2.048	1.979	49.725	0.091
				2−2	2.162	2.090	51.949	0.089
				3−1	2.100	2.030	50.115	0.091
			RSD-S	2×2	2.129	2.058	50.315	0.090
				3×2	**2.220**	**2.146**	**52.867**	0.090
		3	SD	3	2.144	2.038	47.360	0.090
			SpecTr	3×3	2.202	2.093	46.211	0.092
				4×3	2.211	2.102	46.960	0.091
			RSD-C	2−2−2	2.484	2.362	54.127	0.090
				3−1−1	2.311	2.196	49.424	0.090
				4−1−1	2.345	2.229	50.509	0.089
			RSD-S	3×3	2.525	2.400	51.902	0.093
				4×3	**2.650**	**2.519**	**54.496**	0.091
		4	SD	4	2.269	2.122	45.826	0.091
			SpecTr	5×4	2.366	2.212	46.726	0.091
				7×4	2.379	2.225	46.287	0.089
			RSD-C	2−2−2−2	2.701	2.526	52.867	0.093
				5−1−1−1	2.503	2.341	49.052	0.089
				7−1−1−1	2.562	2.396	52.106	0.089
			RSD-S	5×4	2.921	2.732	54.744	0.091
				7×4	**3.023**	**2.827**	**56.318**	0.087
		5	SD	5	2.345	2.158	43.302	0.092
			SpecTr	6×5	2.455	2.259	44.595	0.091
				12×5	2.513	2.312	44.089	0.093
			RSD-C	2−2−2−2−2	2.830	2.604	51.392	0.092
				6−1−1−1−1	2.615	2.407	47.560	0.090
				12−1−1−1−1	2.669	2.456	47.987	0.089
			RSD-S	6×5	3.142	2.891	54.142	0.089
				12×5	**3.397**	**3.126**	**58.208**	0.091

Table 8: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	WMT	0	AR	-	1.000	1.000	37.093	0.377
		2	SD	2	1.639	1.584	41.440	0.379
			SpecTr	2×2	1.664	1.608	40.657	0.379
				3×2	1.673	1.617	41.907	0.378
			RSD-C	2−1	1.739	1.681	43.511	0.378
				2−2	1.813	1.752	43.929	0.375
				3−1	1.784	1.724	44.122	0.378
			RSD-S	2×2	1.786	1.726	44.139	0.378
				3×2	**1.865**	**1.802**	**46.030**	0.379
		3	SD	3	1.747	1.660	40.480	0.376
			SpecTr	3×3	1.783	1.695	39.483	0.377
				4×3	1.791	1.702	39.811	0.374
			RSD-C	2−2−2	1.967	1.870	42.825	0.377
				3−1−1	1.896	1.802	42.228	0.379
				4−1−1	1.918	1.824	41.396	0.376
			RSD-S	3×3	2.009	1.909	43.212	0.377
				4×3	**2.064**	**1.962**	**44.172**	0.378
		4	SD	4	1.815	1.697	37.235	0.380
			SpecTr	5×4	1.870	1.749	38.695	0.377
				7×4	1.884	1.761	38.151	0.380
			RSD-C	2−2−2−2	2.067	1.933	41.411	0.377
				5−1−1−1	2.021	1.889	40.048	0.379
				7−1−1−1	2.054	1.921	41.217	0.376
			RSD-S	5×4	2.217	2.073	43.154	0.378
				7×4	**2.290**	**2.142**	**43.310**	0.375
		5	SD	5	1.861	1.713	35.968	0.375
			SpecTr	6×5	1.936	1.781	36.752	0.376
				12×5	1.994	1.835	37.541	0.371
			RSD-C	2−2−2−2−2	2.142	1.971	40.408	0.376
				6−1−1−1−1	2.093	1.926	38.438	0.376
				12−1−1−1−1	2.155	1.983	38.945	0.377
			RSD-S	6×5	2.335	2.149	40.428	0.375
				12×5	**2.488**	**2.289**	**43.359**	0.375

Table 9: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the Dolly task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	Dolly	0	AR	-	1.000	1.000	37.596	-
		2	SD	2	2.122	2.051	51.492	-
			SpecTr	2×2	2.177	2.104	50.471	-
				3×2	2.215	2.140	49.350	-
			RSD-C	2−1	2.182	2.109	50.732	-
				2−2	2.253	2.178	**52.610**	-
				3−1	2.201	2.127	51.639	-
			RSD-S	2×2	2.239	2.164	50.320	-
				3×2	**2.278**	**2.202**	51.740	-
		3	SD	3	2.429	2.309	51.847	-
			SpecTr	3×3	2.549	2.423	54.051	-
				4×3	2.579	2.451	53.358	-
			RSD-C	2−2−2	2.628	2.498	54.402	-
				3−1−1	2.508	2.384	52.888	-
				4−1−1	2.506	2.382	52.892	-
			RSD-S	3×3	2.660	2.528	54.360	-
				4×3	**2.686**	**2.553**	**55.581**	-
		4	SD	4	2.642	2.470	51.204	-
			SpecTr	5×4	2.853	2.668	52.977	-
				7×4	2.888	2.700	54.500	-
			RSD-C	2−2−2−2	2.914	2.725	**55.146**	-
				5−1−1−1	2.716	2.539	52.369	-
				7−1−1−1	2.728	2.551	53.662	-
			RSD-S	5×4	2.994	2.799	54.032	-
				7×4	**3.005**	**2.810**	54.242	-
		5	SD	5	2.764	2.543	50.163	-
			SpecTr	6×5	3.072	2.826	53.907	-
				12×5	3.153	2.902	**55.039**	-
			RSD-C	2−2−2−2−2	3.125	2.876	54.231	-
				6−1−1−1−1	2.854	2.626	51.919	-
				12−1−1−1−1	2.860	2.631	52.547	-
			RSD-S	6×5	3.221	2.963	53.055	-
				12×5	**3.241**	**2.982**	52.504	-

Table 10: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	XSum	0	AR	-	1.000	1.000	28.727	0.112
		2	SD	2	1.941	1.906	38.799	0.113
			SpecTr	2×2	1.973	1.938	38.368	0.110
				3×2	1.976	1.941	38.896	0.114
			RSD-C	2−1	2.044	2.008	40.215	0.111
				2−2	2.166	2.128	42.176	0.112
				3−1	2.093	2.056	39.946	0.114
			RSD-S	2×2	2.127	2.089	40.412	0.111
				3×2	**2.232**	**2.193**	**42.700**	0.111
		3	SD	3	2.160	2.104	39.082	0.111
			SpecTr	3×3	2.200	2.142	39.496	0.113
				4×3	2.211	2.153	38.870	0.110
			RSD-C	2−2−2	2.476	2.411	42.802	0.112
				3−1−1	2.308	2.247	41.188	0.112
				4−1−1	2.346	2.284	41.058	0.109
			RSD-S	3×3	2.522	2.456	43.542	0.112
				4×3	**2.616**	**2.548**	**45.064**	0.113
		4	SD	4	2.290	2.211	38.922	0.113
			SpecTr	5×4	2.376	2.293	38.781	0.111
				7×4	2.387	2.304	38.827	0.113
			RSD-C	2−2−2−2	2.692	2.598	43.417	0.112
				5−1−1−1	2.510	2.423	41.620	0.110
				7−1−1−1	2.553	2.465	41.016	0.109
			RSD-S	5×4	2.924	2.823	46.864	0.112
				7×4	**3.056**	**2.950**	**48.332**	0.111
		5	SD	5	2.371	2.269	37.278	0.111
			SpecTr	6×5	2.499	2.392	38.423	0.111
				12×5	2.530	2.421	38.113	0.113
			RSD-C	2−2−2−2−2	2.853	2.731	43.183	0.110
				6−1−1−1−1	2.621	2.508	40.949	0.112
				12−1−1−1−1	2.684	2.569	39.693	0.112
			RSD-S	6×5	3.153	3.018	46.345	0.112
				12×5	**3.390**	**3.244**	**48.010**	0.108

Table 11: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	WMT	0	AR	-	1.000	1.000	29.233	0.340
		2	SD	2	1.729	1.699	36.679	0.346
			SpecTr	2×2	1.806	1.774	37.746	0.338
				3×2	1.758	1.727	36.160	0.343
			RSD-C	2−1	1.758	1.727	37.315	0.357
				2−2	1.928	1.894	39.464	0.345
				3−1	1.891	1.858	38.600	0.344
			RSD-S	2×2	1.956	1.922	39.052	0.333
				3×2	**2.023**	**1.987**	**40.458**	0.349
		3	SD	3	1.831	1.783	35.515	0.342
			SpecTr	3×3	1.839	1.791	35.208	0.347
				4×3	1.915	1.865	35.999	0.338
			RSD-C	2−2−2	2.078	2.023	38.741	0.335
				3−1−1	2.115	2.060	39.754	0.336
				4−1−1	2.033	1.980	38.500	0.348
			RSD-S	3×3	2.138	2.082	39.120	0.348
				4×3	**2.271**	**2.212**	**40.370**	0.340
		4	SD	4	1.963	1.895	35.054	0.346
			SpecTr	5×4	2.050	1.979	35.219	0.348
				7×4	2.012	1.943	34.089	0.339
			RSD-C	2−2−2−2	2.314	2.233	39.395	0.342
				5−1−1−1	2.098	2.025	36.062	0.344
				7−1−1−1	2.213	2.136	37.582	0.335
			RSD-S	5×4	2.385	2.302	39.546	0.340
				7×4	**2.629**	**2.538**	**43.600**	0.329
		5	SD	5	2.089	1.999	34.688	0.347
			SpecTr	6×5	2.077	1.988	32.472	0.348
				12×5	2.069	1.980	32.444	0.342
			RSD-C	2−2−2−2−2	2.401	2.298	38.278	0.343
				6−1−1−1−1	2.381	2.278	38.381	0.339
				12−1−1−1−1	2.254	2.157	35.717	0.346
			RSD-S	6×5	2.532	2.423	39.268	0.348
				12×5	**2.683**	**2.568**	**41.290**	0.340

Table 12: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the Dolly task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	Dolly	0	AR	-	1.000	1.000	29.672	-
		2	SD	2	2.103	2.066	42.833	-
			SpecTr	2×2	2.158	2.120	41.752	-
				3×2	2.187	2.148	42.515	-
			RSD-C	2−1	2.163	2.125	42.755	-
				2−2	2.241	2.201	43.217	-
				3−1	2.181	2.143	43.658	-
			RSD-S	2×2	2.230	2.191	43.460	-
				3×2	**2.262**	**2.222**	**45.408**	-
		3	SD	3	2.393	2.330	44.136	-
			SpecTr	3×3	2.513	2.447	43.976	-
				4×3	2.548	2.482	45.106	-
			RSD-C	2−2−2	2.614	2.545	46.587	-
				3−1−1	2.471	2.406	44.776	-
				4−1−1	2.481	2.416	45.310	-
			RSD-S	3×3	2.631	2.562	46.313	-
				4×3	**2.660**	**2.590**	**47.439**	-
		4	SD	4	2.590	2.500	43.673	-
			SpecTr	5×4	2.809	2.711	45.791	-
				7×4	2.841	2.743	46.150	-
			RSD-C	2−2−2−2	2.885	2.785	46.618	-
				5−1−1−1	2.671	2.579	44.789	-
				7−1−1−1	2.684	2.591	44.729	-
			RSD-S	5×4	2.958	2.855	46.212	-
				7×4	**2.976**	**2.873**	**46.711**	-
		5	SD	5	2.710	2.593	42.688	-
			SpecTr	6×5	3.009	2.880	45.217	-
				12×5	3.083	2.951	45.207	-
			RSD-C	2−2−2−2−2	3.059	2.928	44.830	-
				6−1−1−1−1	2.811	2.690	42.811	-
				12−1−1−1−1	2.810	2.690	42.134	-
			RSD-S	6×5	3.172	3.036	**46.303**	-
				12×5	**3.222**	**3.084**	45.254	-

Table 13: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	XSum	0	AR	-	1.000	1.000	9.242	0.118
		2	SD	2	1.905	1.899	14.110	0.121
			SpecTr	2×2	1.933	1.926	14.048	0.121
				3×2	1.939	1.932	14.057	0.122
			RSD-C	2−1	2.017	2.010	14.688	0.118
				2−2	2.130	2.123	15.354	0.118
				3−1	2.074	2.067	14.868	0.118
			RSD-S	2×2	2.093	2.086	15.080	0.119
				3×2	**2.195**	**2.188**	**15.645**	0.119
		3	SD	3	2.098	2.088	14.865	0.120
			SpecTr	3×3	2.159	2.148	14.875	0.121
				4×3	2.163	2.152	14.798	0.120
			RSD-C	2−2−2	2.440	2.427	16.425	0.120
				3−1−1	2.273	2.261	15.561	0.121
				4−1−1	2.295	2.283	15.542	0.119
			RSD-S	3×3	2.478	2.466	16.644	0.121
				4×3	**2.586**	**2.573**	**17.256**	0.120
		4	SD	4	2.204	2.189	14.860	0.120
			SpecTr	5×4	2.302	2.286	14.639	0.119
				7×4	2.319	2.304	14.479	0.121
			RSD-C	2−2−2−2	2.624	2.606	16.203	0.121
				5−1−1−1	2.454	2.437	15.492	0.122
				7−1−1−1	2.482	2.465	15.430	0.121
			RSD-S	5×4	2.854	2.835	17.528	0.122
				7×4	**2.985**	**2.964**	**18.034**	0.120
		5	SD	5	2.289	2.270	14.734	0.123
			SpecTr	6×5	2.412	2.392	14.608	0.120
				12×5	2.439	2.419	14.016	0.119
			RSD-C	2−2−2−2−2	2.728	2.705	15.593	0.121
				6−1−1−1−1	2.549	2.528	15.182	0.120
				12−1−1−1−1	2.619	2.597	15.029	0.118
			RSD-S	6×5	3.068	3.043	17.742	0.117
				12×5	**3.325**	**3.297**	**18.217**	0.121

Table 14: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	WMT	0	AR	-	1.000	1.000	9.754	0.426
		2	SD	2	1.647	1.642	13.305	0.424
			SpecTr	2×2	1.668	1.663	13.144	0.425
				3×2	1.680	1.674	13.218	0.423
			RSD-C	2−1	1.738	1.732	13.796	0.422
				2−2	1.819	1.813	14.305	0.423
				3−1	1.790	1.783	14.044	0.422
			RSD-S	2×2	1.790	1.784	13.995	0.425
				3×2	**1.871**	**1.865**	**14.620**	0.423
		3	SD	3	1.754	1.745	13.420	0.424
			SpecTr	3×3	1.799	1.790	13.407	0.426
				4×3	1.802	1.793	13.346	0.424
			RSD-C	2−2−2	1.980	1.970	14.577	0.424
				3−1−1	1.908	1.898	14.252	0.425
				4−1−1	1.937	1.927	14.338	0.425
			RSD-S	3×3	2.023	2.013	14.938	0.425
				4×3	**2.086**	**2.075**	**15.205**	0.423
		4	SD	4	1.832	1.819	13.440	0.427
			SpecTr	5×4	1.880	1.867	13.031	0.425
				7×4	1.896	1.884	12.960	0.422
			RSD-C	2−2−2−2	2.079	2.065	14.068	0.423
				5−1−1−1	2.034	2.020	13.980	0.420
				7−1−1−1	2.069	2.055	13.867	0.423
			RSD-S	5×4	2.225	2.210	14.970	0.424
				7×4	**2.306**	**2.290**	**15.231**	0.423
		5	SD	5	1.865	1.849	12.976	0.427
			SpecTr	6×5	1.944	1.927	12.627	0.424
				12×5	1.967	1.951	12.359	0.424
			RSD-C	2−2−2−2−2	2.149	2.131	13.449	0.426
				6−1−1−1−1	2.105	2.088	13.736	0.426
				12−1−1−1−1	2.166	2.148	13.496	0.424
			RSD-S	6×5	2.343	2.323	14.732	0.425
				12×5	**2.509**	**2.488**	**15.249**	0.428

Table 15: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the Dolly task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	Dolly	0	AR	-	1.000	1.000	9.718	-
		2	SD	2	2.080	2.073	16.477	-
			SpecTr	2×2	2.134	2.126	16.481	-
				3×2	2.166	2.158	16.682	-
			RSD-C	2−1	2.136	2.129	16.674	-
				2−2	2.218	2.210	**17.281**	-
				3−1	2.153	2.146	16.766	-
			RSD-S	2×2	2.200	2.193	17.141	-
				3×2	**2.241**	**2.234**	17.264	-
		3	SD	3	2.355	2.343	17.809	-
			SpecTr	3×3	2.479	2.467	18.192	-
				4×3	2.508	2.496	18.244	-
			RSD-C	2−2−2	2.573	2.560	18.954	-
				3−1−1	2.431	2.419	18.064	-
				4−1−1	2.444	2.431	18.121	-
			RSD-S	3×3	2.604	2.591	19.036	-
				4×3	**2.632**	**2.618**	**19.177**	-
		4	SD	4	2.538	2.521	18.163	-
			SpecTr	5×4	2.748	2.730	18.691	-
				7×4	2.796	2.777	18.650	-
			RSD-C	2−2−2−2	2.830	2.810	19.677	-
				5−1−1−1	2.626	2.608	18.701	-
				7−1−1−1	2.634	2.616	18.624	-
			RSD-S	5×4	2.905	2.886	19.845	-
				7×4	**2.942**	**2.922**	**20.072**	-
		5	SD	5	2.658	2.635	18.355	-
			SpecTr	6×5	2.958	2.933	18.789	-
				12×5	3.038	3.013	18.532	-
			RSD-C	2−2−2−2−2	3.015	2.990	19.950	-
				6−1−1−1−1	2.748	2.725	18.807	-
				12−1−1−1−1	2.764	2.740	18.725	-
			RSD-S	6×5	3.127	3.101	20.180	-
				12×5	**3.164**	**3.138**	**20.281**	-

Table 16: We summarize experiment results with OPT 13B target and 125M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-13B	XSum	0	AR	-	1.000	1.000	42.367	0.127
		2	SD	2	1.751	1.718	30.824	0.129
			SpecTr	2×2	1.813	1.778	29.711	0.131
				3×2	1.833	1.798	30.132	0.127
			RSD-C	2−1	1.842	1.807	30.599	0.128
				2−2	1.909	1.872	30.851	0.124
				3−1	1.854	1.818	30.008	0.127
			RSD-S	2×2	1.871	1.835	31.408	0.129
				3×2	**1.930**	**1.893**	**31.803**	0.124
		3	SD	3	1.986	1.930	29.710	0.128
			SpecTr	3×3	1.960	1.904	27.323	0.132
				4×3	2.013	1.956	27.824	0.125
			RSD-C	2−2−2	2.126	2.065	29.494	0.127
				3−1−1	2.011	1.954	28.503	0.129
				4−1−1	2.084	2.025	29.138	0.126
			RSD-S	3×3	2.163	2.102	29.968	0.126
				4×3	**2.216**	**2.153**	**30.852**	0.125
		4	SD	4	1.998	1.923	26.381	0.126
			SpecTr	5×4	2.083	2.005	25.273	0.131
				7×4	2.232	2.149	26.990	0.126
			RSD-C	2−2−2−2	2.248	2.164	26.945	0.127
				5−1−1−1	2.203	2.121	26.424	0.125
				7−1−1−1	2.148	2.068	25.358	0.125
			RSD-S	5×4	2.350	2.262	28.305	0.126
				7×4	**2.476**	**2.384**	**29.880**	0.122
		5	SD	5	2.063	1.967	24.052	0.123
			SpecTr	6×5	2.264	2.159	24.458	0.128
				12×5	2.405	2.293	24.413	0.127
			RSD-C	2−2−2−2−2	2.443	2.329	25.559	0.126
				6−1−1−1−1	2.260	2.155	24.876	0.129
				12−1−1−1−1	2.184	2.083	22.646	0.122
			RSD-S	6×5	2.503	2.387	**25.885**	0.124
				12×5	**2.581**	**2.461**	25.507	0.128

Table 17: We summarize experiment results with OPT 13B target and 125M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-13B	WMT	0	AR	-	1.000	1.000	37.028	0.318
		2	SD	2	1.426	1.399	25.706	0.325
			SpecTr	2×2	1.469	1.441	24.168	0.320
				3×2	1.493	1.464	25.004	0.323
			RSD-C	2−1	1.515	1.486	25.733	0.315
				2−2	1.576	1.546	26.510	0.320
				3−1	1.592	1.561	26.555	0.320
			RSD-S	2×2	1.549	1.520	25.100	0.315
				3×2	**1.630**	**1.598**	**26.872**	0.320
		3	SD	3	1.466	1.424	22.810	0.326
			SpecTr	3×3	1.544	1.500	22.404	0.319
				4×3	1.564	1.520	22.173	0.322
			RSD-C	2−2−2	1.658	1.611	23.247	0.317
				3−1−1	1.605	1.559	23.189	0.319
				4−1−1	1.670	1.623	23.911	0.317
			RSD-S	3×3	1.687	1.639	24.159	0.315
				4×3	**1.735**	**1.685**	**24.195**	0.320
		4	SD	4	1.478	1.423	19.810	0.323
			SpecTr	5×4	1.597	1.537	19.865	0.320
				7×4	1.634	1.573	20.351	0.317
			RSD-C	2−2−2−2	1.682	1.619	20.922	0.319
				5−1−1−1	1.670	1.608	20.916	0.319
				7−1−1−1	1.705	1.641	20.649	0.321
			RSD-S	5×4	1.848	1.779	**22.590**	0.314
				7×4	**1.877**	**1.806**	22.264	0.318
		5	SD	5	1.483	1.414	17.405	0.317
			SpecTr	6×5	1.668	1.590	18.722	0.321
				12×5	1.686	1.608	17.612	0.319
			RSD-C	2−2−2−2−2	1.703	1.624	18.167	0.320
				6−1−1−1−1	1.751	1.669	19.170	0.316
				12−1−1−1−1	1.763	1.682	18.373	0.319
			RSD-S	6×5	1.860	1.774	**20.295**	0.316
				12×5	**1.983**	**1.891**	19.838	0.322

Table 18: We summarize experiment results with OPT 30B target and 125M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-30B	XSum	0	AR	-	1.000	1.000	20.438	0.126
		2	SD	2	1.862	1.846	23.711	0.126
			SpecTr	2×2	1.866	1.850	22.566	0.122
				3×2	1.944	1.928	23.188	0.127
			RSD-C	2−1	1.913	1.897	23.400	0.125
				2−2	1.995	1.978	23.434	0.121
				3−1	1.944	1.928	23.315	0.121
			RSD-S	2×2	2.023	2.006	**24.688**	0.122
				3×2	**2.032**	**2.015**	24.074	0.123
		3	SD	3	2.054	2.029	23.173	0.124
			SpecTr	3×3	2.174	2.147	23.102	0.125
				4×3	2.172	2.146	22.684	0.122
			RSD-C	2−2−2	2.236	2.209	23.354	0.121
				3−1−1	2.162	2.135	22.869	0.120
				4−1−1	2.210	2.183	23.294	0.123
			RSD-S	3×3	2.270	2.242	23.394	0.125
				4×3	**2.328**	**2.299**	**24.006**	0.126
		4	SD	4	2.150	2.114	22.063	0.124
			SpecTr	5×4	2.390	2.350	22.741	0.125
				7×4	2.438	2.398	22.644	0.118
			RSD-C	2−2−2−2	2.509	2.468	23.471	0.120
				5−1−1−1	2.358	2.319	22.303	0.123
				7−1−1−1	2.348	2.309	22.029	0.125
			RSD-S	5×4	2.579	2.537	**23.926**	0.122
				7×4	**2.609**	**2.567**	23.362	0.124
		5	SD	5	2.285	2.239	20.944	0.125
			SpecTr	6×5	2.398	2.349	20.216	0.125
				12×5	2.685	2.630	20.243	0.123
			RSD-C	2−2−2−2−2	2.600	2.547	20.488	0.123
				6−1−1−1−1	2.335	2.287	20.014	0.124
				12−1−1−1−1	2.385	2.336	18.672	0.124
			RSD-S	6×5	2.746	2.690	**22.562**	0.121
				12×5	**2.824**	**2.766**	21.108	0.128

Table 19: We summarize experiment results with OPT 30B target and 125M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-30B	WMT	0	AR	-	1.000	1.000	19.180	0.347
		2	SD	2	1.430	1.418	18.274	0.341
			SpecTr	2×2	1.479	1.466	18.092	0.346
				3×2	1.480	1.468	17.717	0.345
			RSD-C	2−1	1.494	1.481	18.121	0.342
				2−2	1.563	1.550	18.484	0.344
				3−1	1.546	1.533	18.216	0.342
			RSD-S	2×2	1.531	1.519	18.386	0.344
				3×2	**1.609**	**1.596**	**18.954**	0.344
		3	SD	3	1.441	1.423	16.582	0.346
			SpecTr	3×3	1.544	1.525	16.561	0.342
				4×3	1.538	1.519	16.183	0.345
			RSD-C	2−2−2	1.623	1.603	17.255	0.343
				3−1−1	1.584	1.564	16.755	0.344
				4−1−1	1.618	1.598	16.873	0.339
			RSD-S	3×3	1.691	1.670	**17.699**	0.343
				4×3	**1.717**	**1.695**	17.429	0.345
		4	SD	4	1.455	1.431	14.959	0.342
			SpecTr	5×4	1.575	1.549	15.176	0.340
				7×4	1.602	1.576	15.264	0.342
			RSD-C	2−2−2−2	1.658	1.631	15.613	0.339
				5−1−1−1	1.665	1.638	15.827	0.344
				7−1−1−1	1.694	1.667	15.815	0.348
			RSD-S	5×4	1.781	1.752	16.587	0.344
				7×4	**1.850**	**1.819**	**16.597**	0.338
		5	SD	5	1.457	1.427	13.614	0.339
			SpecTr	6×5	1.599	1.566	13.906	0.343
				12×5	1.679	1.645	13.389	0.340
			RSD-C	2−2−2−2−2	1.678	1.644	13.557	0.346
				6−1−1−1−1	1.683	1.649	14.351	0.348
				12−1−1−1−1	1.742	1.706	13.892	0.347
			RSD-S	6×5	1.837	1.800	**15.287**	0.338
				12×5	**1.961**	**1.921**	15.112	0.347

Table 20: We summarize experiment results with OPT 66B target and 125M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-66B	XSum	0	AR	-	1.000	1.000	9.550	0.125
		2	SD	2	2.047	2.040	14.726	0.125
			SpecTr	2×2	2.080	2.073	14.598	0.120
				3×2	2.140	2.132	14.638	0.121
			RSD-C	2−1	2.165	2.157	14.885	0.124
				2−2	2.122	2.114	14.755	0.122
				3−1	2.139	2.131	14.896	0.123
			RSD-S	2×2	2.090	2.082	14.578	0.122
				3×2	**2.218**	**2.210**	**15.424**	0.121
		3	SD	3	2.310	2.297	14.920	0.124
			SpecTr	3×3	2.427	2.413	15.373	0.125
				4×3	2.438	2.424	15.477	0.125
			RSD-C	2−2−2	**2.644**	**2.629**	**16.701**	0.119
				3−1−1	2.532	2.517	16.128	0.125
				4−1−1	2.301	2.288	14.111	0.124
			RSD-S	3×3	2.407	2.393	15.239	0.123
				4×3	2.515	2.501	15.800	0.128
		4	SD	4	2.571	2.552	15.668	0.125
			SpecTr	5×4	2.566	2.546	14.881	0.126
				7×4	2.729	2.709	15.539	0.121
			RSD-C	2−2−2−2	2.900	2.878	16.261	0.129
				5−1−1−1	2.715	2.694	15.571	0.124
				7−1−1−1	2.851	2.829	16.362	0.119
			RSD-S	5×4	**2.972**	**2.950**	15.489	0.125
				7×4	2.895	2.873	**16.371**	0.120
		5	SD	5	2.884	2.856	15.897	0.120
			SpecTr	6×5	2.852	2.825	13.819	0.121
				12×5	2.897	2.870	14.488	0.125
			RSD-C	2−2−2−2−2	3.082	3.053	15.712	0.121
				6−1−1−1−1	2.990	2.962	15.533	0.123
				12−1−1−1−1	2.726	2.700	13.769	0.125
			RSD-S	6×5	2.920	2.893	13.903	0.122
				12×5	**3.318**	**3.287**	**16.555**	0.124

Table 21: We summarize experiment results with OPT 66B target and 125M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-66B	WMT	0	AR	-	1.000	1.000	9.418	0.359
		2	SD	2	1.416	1.411	10.206	0.356
			SpecTr	2×2	1.464	1.458	10.209	0.361
				3×2	1.486	1.481	10.285	0.356
			RSD-C	2−1	1.500	1.494	10.371	0.360
				2−2	1.570	1.564	10.613	0.361
				3−1	1.557	1.551	10.846	0.360
			RSD-S	2×2	1.541	1.535	10.628	0.358
				3×2	**1.619**	**1.613**	**11.203**	0.361
		3	SD	3	1.449	1.441	9.594	0.355
			SpecTr	3×3	1.524	1.515	9.574	0.359
				4×3	1.549	1.540	9.896	0.359
			RSD-C	2−2−2	1.635	1.625	10.243	0.361
				3−1−1	1.591	1.582	10.151	0.361
				4−1−1	1.630	1.621	10.329	0.358
			RSD-S	3×3	1.671	1.661	**10.602**	0.359
				4×3	**1.727**	**1.717**	10.565	0.360
		4	SD	4	1.488	1.477	8.982	0.353
			SpecTr	5×4	1.589	1.577	9.166	0.358
				7×4	1.608	1.596	9.280	0.356
			RSD-C	2−2−2−2	1.669	1.656	9.716	0.360
				5−1−1−1	1.664	1.651	9.482	0.362
				7−1−1−1	1.695	1.682	9.657	0.358
			RSD-S	5×4	1.796	1.783	10.292	0.355
				7×4	**1.860**	**1.846**	**10.565**	0.359
		5	SD	5	1.467	1.453	8.353	0.356
			SpecTr	6×5	1.639	1.624	8.803	0.356
				12×5	1.710	1.694	8.709	0.351
			RSD-C	2−2−2−2−2	1.684	1.668	8.693	0.357
				6−1−1−1−1	1.692	1.676	9.046	0.355
				12−1−1−1−1	1.742	1.725	8.817	0.358
			RSD-S	6×5	1.846	1.829	9.503	0.359
				12×5	**1.972**	**1.953**	**9.786**	0.361

Table 22: We summarize experiment results with OPT 13B target and 350M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-13B	XSum	0	AR	-	1.000	1.000	38.088	0.130
		2	SD	2	1.680	1.598	**24.407**	0.131
			SpecTr	2×2	1.724	1.639	23.198	0.125
				3×2	1.727	1.642	22.684	0.132
			RSD-C	2−1	1.703	1.620	22.835	0.127
				2−2	1.793	1.705	23.643	0.125
				3−1	1.739	1.654	23.101	0.125
			RSD-S	2×2	1.713	1.629	22.962	0.126
				3×2	**1.808**	**1.720**	23.932	0.129
		3	SD	3	1.769	1.642	20.340	0.132
			SpecTr	3×3	1.837	1.705	19.272	0.125
				4×3	1.840	1.708	19.607	0.127
			RSD-C	2−2−2	1.965	1.824	21.037	0.125
				3−1−1	1.845	1.712	20.194	0.128
				4−1−1	1.951	1.811	20.799	0.130
			RSD-S	3×3	1.963	1.822	**21.193**	0.125
				4×3	**1.970**	**1.829**	20.446	0.129
		4	SD	4	1.846	1.673	17.508	0.127
			SpecTr	5×4	2.048	1.856	18.112	0.126
				7×4	1.902	1.724	16.552	0.129
			RSD-C	2−2−2−2	1.975	1.791	17.631	0.125
				5−1−1−1	2.015	1.827	18.346	0.123
				7−1−1−1	2.054	1.862	17.945	0.127
			RSD-S	5×4	**2.141**	**1.941**	**18.920**	0.128
				7×4	2.132	1.932	18.778	0.127
		5	SD	5	1.885	1.670	15.230	0.126
			SpecTr	6×5	2.044	1.811	15.505	0.127
				12×5	2.170	1.922	16.132	0.124
			RSD-C	2−2−2−2−2	2.101	1.861	15.743	0.126
				6−1−1−1−1	2.094	1.855	15.918	0.125
				12−1−1−1−1	2.128	1.885	15.723	0.128
			RSD-S	6×5	**2.274**	**2.015**	**17.290**	0.127
				12×5	2.227	1.973	16.334	0.127

Table 23: We summarize experiment results with OPT 13B target and 350M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-13B	WMT	0	AR	-	1.000	1.000	41.276	0.316
		2	SD	2	1.308	1.244	19.246	0.320
			SpecTr	2×2	1.307	1.243	18.186	0.322
				3×2	1.327	1.262	18.285	0.319
			RSD-C	2−1	1.368	1.301	18.828	0.316
				2−2	1.397	1.329	18.816	0.318
				3−1	1.379	1.311	18.856	0.320
			RSD-S	2×2	1.357	1.291	18.703	0.322
				3×2	**1.399**	**1.330**	**19.530**	0.320
		3	SD	3	1.298	1.205	15.259	0.316
			SpecTr	3×3	1.345	1.248	14.972	0.317
				4×3	1.357	1.260	14.815	0.321
			RSD-C	2−2−2	1.395	1.295	15.120	0.315
				3−1−1	1.392	1.292	15.291	0.318
				4−1−1	1.413	1.312	15.581	0.316
			RSD-S	3×3	1.426	1.324	15.505	0.317
				4×3	**1.494**	**1.387**	**16.732**	0.321
		4	SD	4	1.304	1.182	12.579	0.317
			SpecTr	5×4	1.380	1.251	12.382	0.310
				7×4	1.392	1.262	12.599	0.321
			RSD-C	2−2−2−2	1.411	1.280	12.893	0.320
				5−1−1−1	1.431	1.297	13.117	0.317
				7−1−1−1	1.453	1.317	13.195	0.319
			RSD-S	5×4	1.483	1.344	13.532	0.318
				7×4	**1.519**	**1.378**	**13.554**	0.318
		5	SD	5	1.313	1.164	10.823	0.317
			SpecTr	6×5	1.390	1.231	10.736	0.318
				12×5	1.414	1.253	10.731	0.317
			RSD-C	2−2−2−2−2	1.407	1.247	10.742	0.315
				6−1−1−1−1	1.489	1.319	11.523	0.320
				12−1−1−1−1	1.485	1.316	11.321	0.322
			RSD-S	6×5	1.516	1.343	11.630	0.316
				12×5	**1.571**	**1.392**	**11.758**	0.317

Table 24: We summarize experiment results with OPT 30B target and 350M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-30B	XSum	0	AR	-	1.000	1.000	20.116	0.125
		2	SD	2	1.815	1.776	19.846	0.122
			SpecTr	2×2	1.874	1.834	19.436	0.120
				3×2	1.872	1.831	19.006	0.124
			RSD-C	2−1	1.923	1.881	**20.230**	0.125
				2−2	1.952	1.910	19.745	0.122
				3−1	1.872	1.831	18.957	0.124
			RSD-S	2×2	1.941	1.899	20.146	0.123
				3×2	**1.972**	**1.929**	19.953	0.122
		3	SD	3	1.990	1.926	18.017	0.121
			SpecTr	3×3	2.018	1.953	17.299	0.126
				4×3	2.095	2.028	17.858	0.125
			RSD-C	2−2−2	2.082	2.015	17.384	0.124
				3−1−1	2.099	2.032	18.107	0.123
				4−1−1	2.167	2.098	**19.136**	0.122
			RSD-S	3×3	**2.243**	**2.171**	18.895	0.125
				4×3	2.234	2.163	18.645	0.128
		4	SD	4	2.112	2.023	16.732	0.121
			SpecTr	5×4	2.248	2.153	16.436	0.128
				7×4	2.306	2.208	16.922	0.119
			RSD-C	2−2−2−2	2.292	2.195	16.546	0.123
				5−1−1−1	2.220	2.126	16.732	0.128
				7−1−1−1	2.276	2.180	17.192	0.122
			RSD-S	5×4	**2.487**	**2.382**	**18.310**	0.121
				7×4	2.456	2.352	17.676	0.120
		5	SD	5	2.194	2.079	15.020	0.125
			SpecTr	6×5	2.399	2.274	**15.636**	0.124
				12×5	2.349	2.226	14.121	0.125
			RSD-C	2−2−2−2−2	2.279	2.160	13.978	0.123
				6−1−1−1−1	2.318	2.197	15.534	0.122
				12−1−1−1−1	2.251	2.133	14.118	0.118
			RSD-S	6×5	2.420	2.294	15.455	0.125
				12×5	**2.598**	**2.462**	15.547	0.122

Table 25: We summarize experiment results with OPT 30B target and 350M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-30B	WMT	0	AR	-	1.000	1.000	19.107	0.341
		2	SD	2	1.276	1.248	14.344	0.341
			SpecTr	2×2	1.313	1.284	13.685	0.347
				3×2	1.324	1.295	13.659	0.342
			RSD-C	2−1	1.346	1.317	14.331	0.344
				2−2	1.378	1.348	14.080	0.350
				3−1	1.400	1.370	**14.638**	0.340
			RSD-S	2×2	1.360	1.330	14.031	0.345
				3×2	**1.407**	**1.376**	14.399	0.345
		3	SD	3	1.333	1.290	**12.584**	0.348
			SpecTr	3×3	1.348	1.305	11.763	0.347
				4×3	1.363	1.320	11.966	0.339
			RSD-C	2−2−2	1.403	1.358	12.176	0.349
				3−1−1	1.394	1.349	12.406	0.346
				4−1−1	1.410	1.365	12.574	0.341
			RSD-S	3×3	1.429	1.383	12.195	0.343
				4×3	**1.463**	**1.416**	12.469	0.345
		4	SD	4	1.302	1.247	10.412	0.346
			SpecTr	5×4	1.380	1.321	10.553	0.347
				7×4	1.427	1.367	10.761	0.346
			RSD-C	2−2−2−2	1.413	1.353	10.621	0.344
				5−1−1−1	1.436	1.375	10.946	0.344
				7−1−1−1	1.462	1.400	11.024	0.339
			RSD-S	5×4	1.492	1.429	11.074	0.343
				7×4	**1.532**	**1.467**	**11.250**	0.345
		5	SD	5	1.351	1.280	9.465	0.344
			SpecTr	6×5	1.388	1.315	9.235	0.345
				12×5	1.418	1.344	8.974	0.345
			RSD-C	2−2−2−2−2	1.454	1.378	9.056	0.343
				6−1−1−1−1	1.456	1.380	9.977	0.341
				12−1−1−1−1	1.491	1.413	9.556	0.344
			RSD-S	6×5	1.517	1.438	**10.031**	0.342
				12×5	**1.583**	**1.500**	9.751	0.349

Table 26: We summarize experiment results with OPT 66B target and 350M draft for the XSum task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-66B	XSum	0	AR	-	1.000	1.000	9.225	0.123
		2	SD	2	1.923	1.904	12.282	0.122
			SpecTr	2×2	1.999	1.979	12.150	0.124
				3×2	1.932	1.913	11.550	0.124
			RSD-C	2−1	2.067	2.046	**12.628**	0.122
				2−2	2.020	1.999	11.476	0.125
				3−1	2.038	2.018	12.425	0.123
			RSD-S	2×2	2.013	1.993	12.109	0.122
				3×2	**2.070**	**2.049**	12.542	0.126
		3	SD	3	2.223	2.189	12.241	0.121
			SpecTr	3×3	2.278	2.244	11.737	0.122
				4×3	2.153	2.121	11.238	0.121
			RSD-C	2−2−2	2.325	2.290	12.219	0.126
				3−1−1	2.238	2.205	11.145	0.121
				4−1−1	2.238	2.205	11.743	0.124
			RSD-S	3×3	2.402	2.366	12.680	0.125
				4×3	**2.444**	**2.407**	**12.687**	0.121
		4	SD	4	2.346	2.300	11.336	0.123
			SpecTr	5×4	2.476	2.427	11.431	0.124
				7×4	2.611	2.560	11.891	0.128
			RSD-C	2−2−2−2	2.544	2.494	10.932	0.127
				5−1−1−1	2.617	2.566	11.428	0.123
				7−1−1−1	2.677	2.624	12.065	0.123
			RSD-S	5×4	**2.679**	**2.626**	**12.195**	0.121
				7×4	2.660	2.607	12.024	0.119
		5	SD	5	2.603	2.539	11.183	0.123
			SpecTr	6×5	2.652	2.586	10.904	0.123
				12×5	2.742	2.675	10.673	0.119
			RSD-C	2−2−2−2−2	2.724	2.657	10.857	0.119
				6−1−1−1−1	2.587	2.523	10.554	0.124
				12−1−1−1−1	2.678	2.612	10.542	0.129
			RSD-S	6×5	2.937	2.865	11.434	0.119
				12×5	**3.074**	**2.999**	**12.013**	0.125

Table 27: We summarize experiment results with OPT 66B target and 350M draft for the WMT task with various draft lengths. Draft Length (DL) means the (maximum) length for all draft token sequences generated by the draft model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	DL	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-66B	WMT	0	AR	-	1.000	1.000	9.329	0.355
		2	SD	2	1.270	1.257	8.045	0.359
			SpecTr	2×2	1.296	1.284	8.024	0.361
				3×2	1.313	1.300	8.000	0.358
			RSD-C	2−1	1.326	1.313	8.206	0.358
				2−2	1.353	1.339	8.249	0.358
				3−1	1.358	1.344	8.193	0.358
			RSD-S	2×2	1.349	1.335	8.309	0.358
				3×2	**1.390**	**1.376**	**8.556**	0.361
		3	SD	3	1.284	1.265	7.154	0.357
			SpecTr	3×3	1.338	1.318	7.115	0.358
				4×3	1.346	1.326	7.111	0.358
			RSD-C	2−2−2	1.384	1.364	7.307	0.357
				3−1−1	1.374	1.354	7.260	0.361
				4−1−1	1.402	1.381	7.491	0.362
			RSD-S	3×3	1.415	1.394	7.566	0.359
				4×3	**1.442**	**1.420**	**7.662**	0.359
		4	SD	4	1.290	1.264	6.256	0.358
			SpecTr	5×4	1.358	1.331	6.288	0.356
				7×4	1.376	1.349	6.346	0.357
			RSD-C	2−2−2−2	1.396	1.369	6.430	0.363
				5−1−1−1	1.419	1.391	6.596	0.365
				7−1−1−1	1.442	1.414	6.601	0.356
			RSD-S	5×4	1.473	1.443	6.866	0.362
				7×4	**1.506**	**1.476**	**6.892**	0.356
		5	SD	5	1.295	1.263	5.667	0.356
			SpecTr	6×5	1.373	1.340	5.721	0.360
				12×5	1.399	1.365	5.615	0.359
			RSD-C	2−2−2−2−2	1.393	1.359	5.647	0.357
				6−1−1−1−1	1.431	1.396	5.962	0.359
				12−1−1−1−1	1.471	1.435	5.867	0.362
			RSD-S	6×5	1.489	1.453	6.181	0.360
				12×5	**1.555**	**1.516**	**6.183**	0.358

Table 28: We summarize experiment results with Llama 2-7B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-7B	XSum	1	AR	-	1.000	1.000	37.566	0.141
		6	SD	6	**3.087**	**2.796**	53.455	0.142
			SpecTr	2×3	2.577	2.450	54.070	0.141
				3×2	2.279	2.202	53.730	0.139
			RSD-C	2−1−1	2.571	2.444	55.296	0.139
				2−2	2.398	2.317	**57.629**	0.143
				3−1	2.291	2.214	54.449	0.140
			RSD-S	2×3	2.791	2.653	56.179	0.136
				3×2	2.432	2.350	56.053	0.140
		10	SD	10	**3.438**	2.929	44.156	0.141
			SpecTr	2×5	3.030	2.788	54.224	0.138
				5×2	2.339	2.261	55.258	0.145
			RSD-C	2−1−1−1−1	3.021	2.780	54.235	0.138
				2−2−1	2.725	2.590	58.367	0.142
				5−1	2.338	2.259	53.326	0.141
			RSD-S	2×5	3.300	**3.036**	55.787	0.144
				5×2	2.567	2.481	**58.717**	0.140
		14	SD	14	3.565	2.868	37.602	0.143
			SpecTr	2×7	3.296	2.939	50.642	0.142
				7×2	2.357	2.278	55.548	0.141
			RSD-C	2−1−1−1−1−1−1	3.250	2.898	50.267	0.143
				2−2−2	2.868	2.726	**61.873**	0.141
				7−1	2.374	2.294	54.968	0.136
			RSD-S	2×7	**3.586**	**3.198**	53.409	0.140
				7×2	2.618	2.530	58.397	0.140
		21	SD	21	3.677	2.695	30.002	0.142
			SpecTr	3×7	3.494	3.116	51.744	0.138
				7×3	2.755	2.619	58.586	0.139
			RSD-C	3−1−1−1−1−1−1	3.326	2.966	51.650	0.138
				3−2−2	2.951	2.805	61.424	0.142
				7−1−1	2.741	2.605	58.079	0.139
			RSD-S	3×7	**3.918**	**3.494**	57.997	0.142
				7×3	3.168	3.011	**66.555**	0.141
		30	SD	30	3.743	2.462	22.927	0.139
			SpecTr	5×6	3.353	3.037	55.070	0.145
				6×5	3.209	2.953	56.117	0.141
			RSD-C	2−2−2−2	3.205	2.997	61.608	0.142
				5−1−1−1−1−1	3.222	2.918	52.738	0.142
				6−1−1−1−1	3.133	2.883	55.747	0.142
			RSD-S	5×6	**3.959**	**3.585**	62.351	0.138
				6×5	3.811	3.507	**66.069**	0.141

Table 29: We summarize experiment results with Llama 2-7B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-7B	WMT	1	AR	-	1.000	1.000	37.340	0.374
		6	SD	6	1.953	1.768	34.768	0.378
			SpecTr	2×3	1.857	1.765	41.844	0.374
				3×2	1.757	1.698	42.912	0.376
			RSD-C	2−1−1	1.889	1.796	41.272	0.375
				2−2	1.858	1.796	44.210	0.372
				3−1	1.819	1.758	43.938	0.375
			RSD-S	2×3	**1.977**	**1.879**	42.658	0.371
				3×2	1.912	1.847	**45.982**	0.373
		10	SD	10	2.051	1.748	27.755	0.377
			SpecTr	2×5	1.996	1.836	37.212	0.374
				5×2	1.792	1.732	43.474	0.380
			RSD-C	2−1−1−1−1	2.032	1.869	37.614	0.374
				2−2−1	1.984	1.886	44.174	0.378
				5−1	1.882	1.819	44.958	0.376
			RSD-S	2×5	**2.126**	**1.957**	36.921	0.375
				5×2	2.018	1.950	**47.316**	0.376
		14	SD	14	2.075	1.669	22.528	0.370
			SpecTr	2×7	2.085	1.859	32.730	0.375
				7×2	1.813	1.752	42.856	0.375
			RSD-C	2−1−1−1−1−1−1	2.110	1.882	33.053	0.373
				2−2−2	2.033	1.933	45.579	0.372
				7−1	1.919	1.854	45.194	0.377
			RSD-S	2×7	**2.236**	1.994	33.965	0.374
				7×2	2.071	**2.002**	**47.111**	0.380
		21	SD	21	2.114	1.549	17.569	0.376
			SpecTr	3×7	2.135	1.903	34.329	0.375
				7×3	1.968	1.870	43.981	0.375
			RSD-C	3−1−1−1−1−1−1	2.160	1.926	34.656	0.373
				3−2−2	2.131	2.025	**48.313**	0.372
				7−1−1	2.038	1.937	45.098	0.377
			RSD-S	3×7	**2.354**	2.099	37.088	0.374
				7×3	2.285	**2.172**	48.239	0.372
		30	SD	30	2.177	1.431	13.303	0.375
			SpecTr	5×6	2.152	1.949	37.182	0.374
				6×5	2.120	1.951	39.488	0.373
			RSD-C	2−2−2−2	2.152	2.013	**44.115**	0.378
				5−1−1−1−1−1	2.197	1.990	37.424	0.370
				6−1−1−1−1	2.171	1.998	39.623	0.372
			RSD-S	5×6	2.463	2.231	39.472	0.374
				6×5	**2.467**	**2.270**	43.140	0.370

Table 30: We summarize experiment results with Llama 2-13B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-13B	XSum	1	AR	-	1.000	1.000	27.958	0.166
		6	SD	6	**2.947**	**2.796**	43.504	0.163
			SpecTr	2×3	2.492	2.426	43.202	0.162
				3×2	2.212	2.173	41.701	0.165
			RSD-C	2−1−1	2.474	2.409	43.028	0.164
				2−2	2.347	2.305	44.088	0.166
				3−1	2.269	2.229	43.224	0.158
			RSD-S	2×3	2.660	2.590	45.233	0.165
				3×2	2.412	2.370	**45.802**	0.162
		10	SD	10	**3.248**	2.981	38.044	0.160
			SpecTr	2×5	2.891	2.766	43.939	0.165
				5×2	2.273	2.233	42.925	0.163
			RSD-C	2−1−1−1−1	2.868	2.745	43.935	0.164
				2−2−1	2.655	2.586	46.486	0.165
				5−1	2.312	2.272	44.677	0.163
			RSD-S	2×5	3.139	**3.004**	45.129	0.165
				5×2	2.509	2.465	**47.010**	0.165
		14	SD	14	3.316	2.945	31.464	0.166
			SpecTr	2×7	3.192	3.003	42.485	0.166
				7×2	2.293	2.253	42.888	0.160
			RSD-C	2−1−1−1−1−1−1	3.155	2.968	41.432	0.159
				2−2−2	2.784	2.711	**49.658**	0.166
				7−1	2.316	2.275	43.455	0.168
			RSD-S	2×7	**3.364**	**3.165**	41.883	0.162
				7×2	2.583	2.538	47.255	0.162
		21	SD	21	3.470	2.920	25.644	0.166
			SpecTr	3×7	3.325	3.128	43.834	0.161
				7×3	2.685	2.615	46.056	0.157
			RSD-C	3−1−1−1−1−1−1	3.172	2.984	41.657	0.163
				3−2−2	2.858	2.783	49.377	0.166
				7−1−1	2.629	2.560	44.732	0.161
			RSD-S	3×7	**3.621**	**3.407**	45.864	0.173
				7×3	3.066	2.985	**50.916**	0.165
		30	SD	30	3.589	2.827	20.535	0.164
			SpecTr	5×6	3.334	3.163	45.472	0.165
				6×5	3.108	2.974	45.765	0.169
			RSD-C	2−2−2−2	3.096	2.989	49.061	0.167
				5−1−1−1−1−1	3.125	2.965	44.662	0.165
				6−1−1−1−1	3.014	2.885	45.140	0.163
			RSD-S	5×6	**3.741**	**3.550**	48.762	0.163
				6×5	3.648	3.492	**51.554**	0.162

Table 31: We summarize experiment results with Llama 2-13B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-13B	WMT	1	AR	-	1.000	1.000	28.882	0.413
		6	SD	6	1.950	1.850	30.533	0.409
			SpecTr	2×3	1.844	1.796	33.376	0.411
				3×2	1.748	1.717	36.005	0.408
			RSD-C	2−1−1	1.884	1.834	35.064	0.411
				2−2	1.852	1.819	**37.453**	0.407
				3−1	1.815	1.783	37.156	0.408
			RSD-S	2×3	**1.956**	**1.905**	37.177	0.409
				3×2	1.903	1.869	37.195	0.410
		10	SD	10	2.061	1.891	25.104	0.408
			SpecTr	2×5	1.989	1.904	31.919	0.409
				5×2	1.780	1.749	36.296	0.411
			RSD-C	2−1−1−1−1	2.012	1.926	32.918	0.409
				2−2−1	1.970	1.919	36.758	0.408
				5−1	1.872	1.838	37.063	0.414
			RSD-S	2×5	**2.122**	**2.031**	33.454	0.408
				5×2	2.004	1.968	**39.922**	0.406
		14	SD	14	2.118	1.881	21.448	0.408
			SpecTr	2×7	2.084	1.961	29.443	0.411
				7×2	1.801	1.769	35.500	0.407
			RSD-C	2−1−1−1−1−1−1	2.099	1.975	29.876	0.411
				2−2−2	2.027	1.974	37.232	0.407
				7−1	1.912	1.878	38.123	0.409
			RSD-S	2×7	**2.225**	**2.093**	29.851	0.411
				7×2	2.069	2.032	**40.692**	0.405
		21	SD	21	2.239	1.884	17.204	0.409
			SpecTr	3×7	2.133	2.007	30.245	0.408
				7×3	1.953	1.901	36.695	0.409
			RSD-C	3−1−1−1−1−1−1	2.156	2.028	30.470	0.408
				3−2−2	2.121	2.065	39.876	0.414
				7−1−1	2.041	1.988	37.111	0.411
			RSD-S	3×7	**2.353**	2.214	31.571	0.406
				7×3	2.281	**2.221**	**41.848**	0.407
		30	SD	30	2.341	1.844	13.847	0.409
			SpecTr	5×6	2.153	2.043	32.312	0.407
				6×5	2.101	2.011	34.170	0.408
			RSD-C	2−2−2−2	2.122	2.048	36.484	0.407
				5−1−1−1−1−1	2.190	2.078	32.439	0.408
				6−1−1−1−1	2.163	2.070	34.071	0.408
			RSD-S	5×6	**2.450**	2.325	34.302	0.407
				6×5	2.448	**2.343**	**37.229**	0.408

Table 32: We summarize experiment results with Llama 2-70B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-70B	XSum	1	AR	-	1.000	1.000	9.016	0.194
		6	SD	6	**2.820**	**2.791**	16.776	0.193
			SpecTr	2×3	2.447	2.435	16.240	0.194
				3×2	2.204	2.197	15.335	0.191
			RSD-C	2−1−1	2.475	2.462	16.405	0.192
				2−2	2.322	2.314	16.063	0.189
				3−1	2.239	2.231	15.564	0.197
			RSD-S	2×3	2.625	2.611	**17.100**	0.193
				3×2	2.376	2.368	16.267	0.193
		10	SD	10	**3.142**	**3.090**	16.134	0.192
			SpecTr	2×5	2.836	2.812	17.119	0.192
				5×2	2.235	2.227	15.404	0.198
			RSD-C	2−1−1−1−1	2.829	2.805	17.057	0.194
				2−2−1	2.617	2.604	17.193	0.192
				5−1	2.273	2.266	15.541	0.195
			RSD-S	2×5	3.028	3.003	**17.674**	0.187
				5×2	2.484	2.475	16.683	0.193
		14	SD	14	3.178	3.104	14.472	0.199
			SpecTr	2×7	3.138	3.101	17.512	0.191
				7×2	2.262	2.254	15.409	0.194
			RSD-C	2−1−1−1−1−1−1	3.028	2.993	16.938	0.189
				2−2−2	2.757	2.743	17.722	0.188
				7−1	2.296	2.288	15.587	0.193
			RSD-S	2×7	**3.311**	**3.272**	**17.882**	0.192
				7×2	2.565	2.556	17.094	0.191
		21	SD	21	3.321	3.207	12.428	0.193
			SpecTr	3×7	3.160	3.123	17.143	0.188
				7×3	2.633	2.620	16.486	0.188
			RSD-C	3−1−1−1−1−1−1	3.104	3.068	16.920	0.192
				3−2−2	2.837	2.822	17.726	0.188
				7−1−1	2.579	2.566	16.201	0.192
			RSD-S	3×7	**3.505**	**3.464**	18.299	0.195
				7×3	3.037	3.021	**18.568**	0.188
		30	SD	30	3.341	3.179	10.399	0.189
			SpecTr	5×6	3.213	3.181	17.584	0.189
				6×5	3.103	3.077	17.654	0.192
			RSD-C	2−2−2−2	3.028	3.008	17.863	0.191
				5−1−1−1−1−1	3.074	3.043	17.130	0.197
				6−1−1−1−1	2.935	2.910	16.871	0.193
			RSD-S	5×6	**3.607**	**3.571**	18.880	0.191
				6×5	3.556	3.526	**19.501**	0.192

Table 33: We summarize experiment results with Llama 2-70B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-70B	WMT	1	AR	-	1.000	1.000	9.764	0.439
		6	SD	6	1.955	1.936	13.160	0.441
			SpecTr	2×3	1.856	1.847	13.951	0.437
				3×2	1.756	1.750	13.806	0.445
			RSD-C	2−1−1	1.889	1.880	14.243	0.439
				2−2	1.853	1.847	14.584	0.443
				3−1	1.819	1.813	14.200	0.440
			RSD-S	2×3	**1.963**	**1.953**	14.548	0.441
				3×2	1.907	1.900	**14.744**	0.439
		10	SD	10	2.045	2.011	11.774	0.437
			SpecTr	2×5	1.994	1.977	13.664	0.438
				5×2	1.785	1.779	13.860	0.439
			RSD-C	2−1−1−1−1	2.021	2.004	13.891	0.439
				2−2−1	1.973	1.963	14.671	0.442
				5−1	1.881	1.874	14.520	0.443
			RSD-S	2×5	**2.127**	**2.109**	14.355	0.438
				5×2	2.017	2.011	**15.396**	0.438
		14	SD	14	2.084	2.035	10.407	0.439
			SpecTr	2×7	2.078	2.053	13.221	0.439
				7×2	1.811	1.805	13.954	0.438
			RSD-C	2−1−1−1−1−1−1	2.110	2.085	13.308	0.438
				2−2−2	2.021	2.010	14.861	0.438
				7−1	1.917	1.910	14.658	0.442
			RSD-S	2×7	**2.226**	**2.200**	13.788	0.444
				7×2	2.098	2.091	**15.865**	0.437
		21	SD	21	2.152	2.078	8.847	0.439
			SpecTr	3×7	2.133	2.108	13.207	0.442
				7×3	1.953	1.943	13.914	0.438
			RSD-C	3−1−1−1−1−1−1	2.160	2.135	13.317	0.438
				3−2−2	2.158	2.147	15.302	0.440
				7−1−1	2.043	2.033	14.583	0.439
			RSD-S	3×7	**2.353**	**2.325**	14.047	0.438
				7×3	2.285	2.274	**15.904**	0.437
		30	SD	30	2.252	2.143	7.616	0.443
			SpecTr	5×6	2.150	2.128	13.466	0.439
				6×5	2.113	2.095	13.811	0.443
			RSD-C	2−2−2−2	2.130	2.116	14.395	0.440
				5−1−1−1−1−1	2.193	2.171	13.754	0.437
				6−1−1−1−1	2.173	2.154	14.166	0.440
			RSD-S	5×6	**2.467**	**2.442**	15.083	0.439
				6×5	2.451	2.430	**15.474**	0.439

Table 34: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	XSum	1	AR	-	1.000	1.000	36.326	0.092
		6	SD	6	**2.393**	2.168	40.649	0.092
			SpecTr	2×3	2.177	2.069	46.704	0.090
				3×2	1.972	1.906	46.963	0.092
			RSD-C	2−1−1	2.251	2.140	48.942	0.091
				2−2	2.162	2.090	50.120	0.089
				3−1	2.100	2.030	49.313	0.091
			RSD-S	2×3	2.390	**2.272**	49.905	0.089
				3×2	2.220	2.146	**51.233**	0.090
		10	SD	10	2.531	2.157	32.934	0.088
			SpecTr	2×5	2.403	2.211	44.360	0.091
				5×2	1.993	1.926	47.010	0.089
			RSD-C	2−1−1−1−1	2.470	2.273	44.878	0.091
				2−2−1	2.370	2.253	50.696	0.091
				5−1	2.154	2.082	50.306	0.089
			RSD-S	2×5	**2.635**	**2.424**	45.726	0.091
				5×2	2.360	2.281	**55.248**	0.092
		14	SD	14	2.551	2.052	27.493	0.091
			SpecTr	2×7	2.514	2.241	39.394	0.091
				7×2	1.991	1.924	45.588	0.092
			RSD-C	2−1−1−1−1−1−1	2.567	2.289	40.884	0.091
				2−2−2	2.484	2.362	51.459	0.090
				7−1	2.182	2.108	50.829	0.091
			RSD-S	2×7	**2.781**	**2.480**	41.993	0.092
				7×2	2.417	2.336	**55.556**	0.092
		21	SD	21	2.556	1.873	20.512	0.090
			SpecTr	3×7	2.534	2.260	39.551	0.091
				7×3	2.243	2.132	49.332	0.091
			RSD-C	3−1−1−1−1−1−1	2.643	2.357	42.535	0.090
				3−2−2	2.568	2.441	55.795	0.091
				7−1−1	2.404	2.285	52.558	0.090
			RSD-S	3×7	**3.009**	**2.683**	44.655	0.092
				7×3	2.791	2.653	**58.627**	0.091
		30	SD	30	2.572	1.691	15.946	0.090
			SpecTr	5×6	2.534	2.295	42.130	0.091
				6×5	2.455	2.259	44.572	0.091
			RSD-C	2−2−2−2	2.701	2.526	**53.578**	0.093
				5−1−1−1−1−1	2.654	2.404	45.281	0.090
				6−1−1−1−1	2.615	2.407	47.230	0.090
			RSD-S	5×6	**3.168**	2.869	49.821	0.089
				6×5	3.142	**2.891**	53.573	0.089

Table 35: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	WMT	1	AR	-	1.000	1.000	36.695	0.377
		6	SD	6	1.900	1.721	33.723	0.379
			SpecTr	2×3	1.770	1.683	38.953	0.377
				3×2	1.673	1.617	41.570	0.378
			RSD-C	2−1−1	1.854	1.762	41.448	0.371
				2−2	1.813	1.752	44.864	0.375
				3−1	1.784	1.724	44.721	0.378
			RSD-S	2×3	**1.909**	**1.814**	41.220	0.379
				3×2	1.865	1.802	**44.982**	0.379
		10	SD	10	1.955	1.666	26.231	0.373
			SpecTr	2×5	1.889	1.738	34.516	0.376
				5×2	1.687	1.631	41.615	0.377
			RSD-C	2−1−1−1−1	1.965	1.808	37.284	0.376
				2−2−1	1.920	1.825	43.100	0.380
				5−1	1.838	1.776	44.881	0.381
			RSD-S	2×5	**2.092**	**1.925**	36.867	0.377
				5×2	1.958	1.892	**47.296**	0.376
		14	SD	14	1.961	1.578	21.506	0.375
			SpecTr	2×7	1.958	1.746	31.680	0.378
				7×2	1.695	1.638	42.528	0.376
			RSD-C	2−1−1−1−1−1−1	2.027	1.808	32.022	0.376
				2−2−2	1.967	1.870	43.105	0.377
				7−1	1.867	1.804	44.714	0.378
			RSD-S	2×7	**2.128**	1.898	32.654	0.378
				7×2	2.004	**1.937**	**46.335**	0.379
		21	SD	21	1.965	1.440	16.321	0.376
			SpecTr	3×7	1.975	1.761	31.874	0.377
				7×3	1.808	1.719	39.780	0.376
			RSD-C	3−1−1−1−1−1−1	2.080	1.855	32.993	0.376
				3−2−2	2.066	1.964	45.320	0.377
				7−1−1	1.976	1.878	44.574	0.378
			RSD-S	3×7	**2.241**	1.998	33.997	0.377
				7×3	2.189	**2.080**	**46.892**	0.376
		30	SD	30	1.976	1.300	12.286	0.376
			SpecTr	5×6	1.969	1.784	33.346	0.374
				6×5	1.936	1.781	35.935	0.376
			RSD-C	2−2−2−2	2.067	1.933	**42.206**	0.377
				5−1−1−1−1−1	2.106	1.908	36.939	0.379
				6−1−1−1−1	2.093	1.926	38.918	0.376
			RSD-S	5×6	**2.341**	2.120	37.980	0.378
				6×5	2.335	**2.149**	41.775	0.375

Table 36: We summarize experiment results with Llama 2-Chat-7B target and 115M draft for the Dolly task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-7B	Dolly	1	AR	-	1.000	1.000	37.816	-
		6	SD	6	**2.872**	**2.601**	47.582	-
			SpecTr	2×3	2.510	2.385	52.609	-
				3×2	2.215	2.140	52.020	-
			RSD-C	2−1−1	2.491	2.367	53.526	-
				2−2	2.253	2.178	52.910	-
				3−1	2.201	2.127	52.182	-
			RSD-S	2×3	2.598	2.470	**53.906**	-
				3×2	2.278	2.202	51.508	-
		10	SD	10	**3.077**	2.622	40.373	-
			SpecTr	2×5	2.898	2.666	49.652	-
				5×2	2.230	2.155	50.708	-
			RSD-C	2−1−1−1−1	2.837	2.611	52.227	-
				2−2−1	2.572	2.445	**53.858**	-
				5−1	2.202	2.128	51.832	-
			RSD-S	2×5	3.026	**2.785**	48.969	-
				5×2	2.299	2.222	52.744	-
		14	SD	14	3.133	2.521	33.603	-
			SpecTr	2×7	3.085	2.751	47.101	-
				7×2	2.248	2.172	49.841	-
			RSD-C	2−1−1−1−1−1−1	3.022	2.695	46.031	-
				2−2−2	2.628	2.498	**56.076**	-
				7−1	2.205	2.131	51.055	-
			RSD-S	2×7	**3.244**	**2.892**	44.446	-
				7×2	2.299	2.222	53.245	-
		21	SD	21	3.136	2.298	25.282	-
			SpecTr	3×7	3.180	2.836	48.530	-
				7×3	2.617	2.488	55.692	-
			RSD-C	3−1−1−1−1−1−1	3.031	2.703	46.084	-
				3−2−2	2.659	2.527	**56.135**	-
				7−1−1	2.506	2.382	53.688	-
			RSD-S	3×7	**3.359**	**2.996**	46.775	-
				7×3	2.703	2.569	54.088	-
		30	SD	30	3.158	2.077	18.689	-
			SpecTr	5×6	3.186	2.886	50.516	-
				6×5	3.072	2.826	52.096	-
			RSD-C	2−2−2−2	2.914	2.725	**54.514**	-
				5−1−1−1−1−1	2.975	2.695	48.944	-
				6−1−1−1−1	2.854	2.626	50.489	-
			RSD-S	5×6	**3.334**	**3.020**	49.805	-
				6×5	3.221	2.963	52.867	-

Table 37: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	XSum	1	AR	-	1.000	1.000	28.370	0.112
		6	SD	6	**2.463**	**2.337**	36.486	0.114
			SpecTr	2×3	2.179	2.121	40.043	0.113
				3×2	1.976	1.941	38.373	0.114
			RSD-C	2−1−1	2.255	2.195	40.247	0.111
				2−2	2.166	2.128	41.385	0.112
				3−1	2.093	2.056	40.980	0.114
			RSD-S	2×3	2.400	**2.337**	41.334	0.113
				3×2	2.232	2.193	**42.572**	0.111
		10	SD	10	2.578	2.365	30.909	0.110
			SpecTr	2×5	2.424	2.319	37.801	0.112
				5×2	2.000	1.965	38.179	0.111
			RSD-C	2−1−1−1−1	2.498	2.391	38.946	0.112
				2−2−1	2.385	2.322	42.166	0.109
				5−1	2.153	2.115	42.599	0.112
			RSD-S	2×5	**2.682**	**2.567**	39.875	0.114
				5×2	2.341	2.300	**44.695**	0.111
		14	SD	14	2.652	2.356	26.277	0.111
			SpecTr	2×7	2.581	2.428	35.302	0.112
				7×2	2.003	1.968	38.576	0.110
			RSD-C	2−1−1−1−1−1−1	2.610	2.456	35.417	0.110
				2−2−2	2.476	2.411	44.017	0.112
				7−1	2.183	2.145	41.901	0.112
			RSD-S	2×7	**2.830**	**2.663**	36.540	0.114
				7×2	2.405	2.363	**44.631**	0.114
		21	SD	21	2.646	2.226	20.028	0.112
			SpecTr	3×7	2.598	2.444	35.767	0.112
				7×3	2.239	2.180	39.758	0.112
			RSD-C	3−1−1−1−1−1−1	2.703	2.543	37.079	0.108
				3−2−2	2.589	2.521	46.647	0.110
				7−1−1	2.403	2.340	42.168	0.112
			RSD-S	3×7	**3.016**	**2.838**	39.531	0.109
				7×3	2.784	2.711	**48.602**	0.111
		30	SD	30	2.699	2.126	15.701	0.113
			SpecTr	5×6	2.566	2.435	35.917	0.112
				6×5	2.499	2.392	38.339	0.111
			RSD-C	2−2−2−2	2.692	2.598	44.569	0.112
				5−1−1−1−1−1	2.699	2.561	38.522	0.113
				6−1−1−1−1	2.621	2.508	40.221	0.112
			RSD-S	5×6	**3.201**	**3.037**	43.176	0.110
				6×5	3.153	3.018	**45.666**	0.112

Table 38: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	WMT	1	AR	-	1.000	1.000	28.662	0.340
		6	SD	6	2.060	1.955	31.459	0.342
			SpecTr	2×3	1.895	1.845	35.835	0.330
				3×2	1.758	1.727	36.220	0.343
			RSD-C	2−1−1	2.018	1.965	38.230	0.346
				2−2	1.928	1.894	39.278	0.345
				3−1	1.891	1.858	38.251	0.344
			RSD-S	2×3	**2.108**	**2.053**	38.436	0.335
				3×2	2.023	1.987	**40.327**	0.349
		10	SD	10	**2.253**	2.067	27.166	0.347
			SpecTr	2×5	1.959	1.874	31.367	0.346
				5×2	1.826	1.794	36.691	0.341
			RSD-C	2−1−1−1−1	2.123	2.032	34.561	0.336
				2−2−1	2.117	2.061	40.143	0.343
				5−1	1.870	1.837	38.913	0.346
			RSD-S	2×5	2.222	**2.127**	34.665	0.343
				5×2	2.058	2.022	**40.616**	0.341
		14	SD	14	2.282	2.027	22.844	0.342
			SpecTr	2×7	2.045	1.924	28.418	0.347
				7×2	1.715	1.685	34.563	0.343
			RSD-C	2−1−1−1−1−1−1	2.095	1.971	29.890	0.347
				2−2−2	2.078	2.023	39.021	0.335
				7−1	1.978	1.943	39.510	0.343
			RSD-S	2×7	**2.340**	**2.202**	31.472	0.343
				7×2	2.170	2.132	**41.309**	0.346
		21	SD	21	2.214	1.863	17.282	0.349
			SpecTr	3×7	2.137	2.010	30.289	0.338
				7×3	2.053	1.999	38.693	0.338
			RSD-C	3−1−1−1−1−1−1	2.209	2.078	31.428	0.333
				3−2−2	2.280	2.220	**41.973**	0.337
				7−1−1	2.130	2.075	40.257	0.337
			RSD-S	3×7	**2.442**	**2.297**	33.291	0.354
				7×3	2.280	2.220	41.291	0.346
		30	SD	30	2.223	1.751	13.360	0.338
			SpecTr	5×6	2.208	2.095	32.564	0.340
				6×5	2.077	1.988	33.949	0.348
			RSD-C	2−2−2−2	2.314	2.233	**40.778**	0.342
				5−1−1−1−1−1	2.218	2.104	33.518	0.345
				6−1−1−1−1	2.381	2.278	37.944	0.339
			RSD-S	5×6	2.525	2.396	36.318	0.344
				6×5	**2.532**	**2.423**	38.549	0.348

Table 39: We summarize experiment results with Llama 2-Chat-13B target and 115M draft for the Dolly task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-13B	Dolly	1	AR	-	1.000	1.000	29.385	-
		6	SD	6	**2.832**	**2.687**	41.513	-
			SpecTr	2×3	2.478	2.413	43.632	-
				3×2	2.187	2.148	42.975	-
			RSD-C	2−1−1	2.456	2.392	45.481	-
				2−2	2.241	2.201	44.034	-
				3−1	2.181	2.143	42.823	-
			RSD-S	2×3	2.573	2.505	**45.570**	-
				3×2	2.262	2.222	43.819	-
		10	SD	10	2.978	2.733	35.013	-
			SpecTr	2×5	2.847	2.725	43.931	-
				5×2	2.214	2.175	43.286	-
			RSD-C	2−1−1−1−1	2.781	2.662	42.729	-
				2−2−1	2.533	2.467	**45.282**	-
				5−1	2.178	2.139	41.735	-
			RSD-S	2×5	**2.992**	**2.863**	43.546	-
				5×2	2.287	2.247	44.563	-
		14	SD	14	3.027	2.688	29.490	-
			SpecTr	2×7	3.028	2.849	40.315	-
				7×2	2.234	2.195	42.190	-
			RSD-C	2−1−1−1−1−1−1	2.936	2.762	39.956	-
				2−2−2	2.614	2.545	**48.098**	-
				7−1	2.187	2.148	43.185	-
			RSD-S	2×7	**3.207**	**3.017**	40.385	-
				7×2	2.295	2.254	44.997	-
		21	SD	21	3.052	2.568	22.670	-
			SpecTr	3×7	3.103	2.920	41.949	-
				7×3	2.595	2.527	46.633	-
			RSD-C	3−1−1−1−1−1−1	2.961	2.786	40.646	-
				3−2−2	2.634	2.565	46.239	-
				7−1−1	2.481	2.416	45.767	-
			RSD-S	3×7	**3.302**	**3.106**	39.684	-
				7×3	2.690	2.620	**47.622**	-
		30	SD	30	3.030	2.387	17.571	-
			SpecTr	5×6	3.124	2.964	43.855	-
				6×5	3.009	2.880	45.170	-
			RSD-C	2−2−2−2	2.885	2.785	**47.105**	-
				5−1−1−1−1−1	2.908	2.759	41.746	-
				6−1−1−1−1	2.811	2.690	44.518	-
			RSD-S	5×6	**3.296**	**3.127**	44.636	-
				6×5	3.172	3.036	45.991	-

Table 40: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	XSum	1	AR	-	1.000	1.000	9.177	0.118
		6	SD	6	**2.349**	2.326	14.440	0.122
			SpecTr	2×3	2.133	2.122	14.685	0.121
				3×2	1.939	1.932	14.080	0.122
			RSD-C	2−1−1	2.210	2.199	15.242	0.119
				2−2	2.130	2.123	15.332	0.118
				3−1	2.074	2.067	14.949	0.118
			RSD-S	2×3	2.341	**2.329**	**15.936**	0.123
				3×2	2.195	2.188	15.543	0.119
		10	SD	10	2.441	2.401	13.092	0.122
			SpecTr	2×5	2.347	2.328	14.802	0.121
				5×2	1.958	1.951	14.033	0.120
			RSD-C	2−1−1−1−1	2.412	2.391	15.092	0.121
				2−2−1	2.329	2.318	15.889	0.119
				5−1	2.128	2.120	15.127	0.119
			RSD-S	2×5	**2.597**	**2.575**	16.050	0.119
				5×2	2.316	2.308	**16.253**	0.118
		14	SD	14	2.462	2.405	11.628	0.121
			SpecTr	2×7	2.432	2.404	14.246	0.121
				7×2	1.969	1.962	14.026	0.121
			RSD-C	2−1−1−1−1−1−1	2.496	2.467	14.635	0.119
				2−2−2	2.440	2.427	16.457	0.120
				7−1	2.161	2.154	15.136	0.120
			RSD-S	2×7	**2.709**	**2.677**	15.339	0.120
				7×2	2.379	2.371	**16.484**	0.120
		21	SD	21	2.482	2.397	9.615	0.119
			SpecTr	3×7	2.470	2.442	14.000	0.119
				7×3	2.181	2.170	14.416	0.120
			RSD-C	3−1−1−1−1−1−1	2.570	2.540	14.676	0.121
				3−2−2	2.518	2.506	16.453	0.121
				7−1−1	2.352	2.340	15.425	0.118
			RSD-S	3×7	**2.907**	**2.873**	16.040	0.119
				7×3	2.746	2.732	**17.593**	0.121
		30	SD	30	2.489	2.369	8.037	0.122
			SpecTr	5×6	2.446	2.421	14.147	0.120
				6×5	2.412	2.392	14.499	0.120
			RSD-C	2−2−2−2	2.624	2.606	16.269	0.121
				5−1−1−1−1−1	2.588	2.562	14.991	0.120
				6−1−1−1−1	2.549	2.528	15.230	0.120
			RSD-S	5×6	**3.105**	**3.074**	17.392	0.119
				6×5	3.068	3.043	**17.821**	0.117

Table 41: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	WMT	1	AR	-	1.000	1.000	9.714	0.426
		6	SD	6	1.906	1.887	12.774	0.426
			SpecTr	2×3	1.785	1.776	13.429	0.424
				3×2	1.680	1.674	13.277	0.423
			RSD-C	2−1−1	1.853	1.844	13.911	0.422
				2−2	1.819	1.813	14.266	0.423
				3−1	1.790	1.783	14.097	0.422
			RSD-S	2×3	**1.924**	**1.914**	14.252	0.426
				3×2	1.871	1.865	**14.538**	0.423
		10	SD	10	1.946	1.914	11.272	0.424
			SpecTr	2×5	1.905	1.889	13.154	0.424
				5×2	1.690	1.684	13.147	0.424
			RSD-C	2−1−1−1−1	1.968	1.952	13.602	0.423
				2−2−1	1.929	1.920	14.448	0.425
				5−1	1.844	1.838	14.304	0.425
			RSD-S	2×5	**2.064**	**2.047**	13.926	0.423
				5×2	1.962	1.955	**14.995**	0.426
		14	SD	14	1.951	1.906	9.828	0.425
			SpecTr	2×7	1.957	1.934	12.441	0.421
				7×2	1.700	1.694	13.117	0.422
			RSD-C	2−1−1−1−1−1−1	2.023	2.000	12.924	0.423
				2−2−2	1.980	1.970	14.630	0.424
				7−1	1.873	1.867	14.313	0.425
			RSD-S	2×7	**2.125**	**2.100**	13.136	0.422
				7×2	2.014	2.008	**15.222**	0.421
		21	SD	21	1.955	1.887	7.945	0.422
			SpecTr	3×7	1.977	1.954	12.241	0.425
				7×3	1.822	1.813	13.003	0.425
			RSD-C	3−1−1−1−1−1−1	2.081	2.056	12.696	0.424
				3−2−2	2.079	2.068	14.773	0.426
				7−1−1	1.998	1.988	14.206	0.424
			RSD-S	3×7	**2.245**	**2.219**	13.532	0.426
				7×3	2.203	2.192	**15.415**	0.421
		30	SD	30	1.954	1.859	6.552	0.426
			SpecTr	5×6	1.971	1.951	12.467	0.423
				6×5	1.944	1.927	12.707	0.424
			RSD-C	2−2−2−2	2.079	2.065	14.062	0.423
				5−1−1−1−1−1	2.121	2.100	13.351	0.427
				6−1−1−1−1	2.105	2.088	13.614	0.426
			RSD-S	5×6	**2.354**	**2.331**	14.396	0.427
				6×5	2.343	2.323	**14.884**	0.425

Table 42: We summarize experiment results with Llama 2-Chat-70B target and 115M draft for the Dolly task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
Llama 2-Chat-70B	Dolly	1	AR	-	1.000	1.000	9.741	-
		6	SD	6	**2.738**	**2.710**	18.155	-
			SpecTr	2×3	2.431	2.419	17.907	-
				3×2	2.166	2.158	16.663	-
			RSD-C	2−1−1	2.417	2.405	18.018	-
				2−2	2.218	2.210	17.254	-
				3−1	2.153	2.146	16.782	-
			RSD-S	2×3	2.545	2.532	**18.573**	-
				3×2	2.241	2.234	17.309	-
		10	SD	10	2.873	2.825	16.293	-
			SpecTr	2×5	2.780	2.757	18.749	-
				5×2	2.198	2.191	16.720	-
			RSD-C	2−1−1−1−1	2.720	2.697	18.665	-
				2−2−1	2.501	2.488	18.443	-
				5−1	2.166	2.158	16.850	-
			RSD-S	2×5	**2.916**	**2.891**	**19.218**	-
				5×2	2.270	2.262	17.558	-
		14	SD	14	2.916	2.849	14.255	-
			SpecTr	2×7	2.951	2.916	18.354	-
				7×2	2.210	2.202	16.724	-
			RSD-C	2−1−1−1−1−1−1	2.878	2.845	18.090	-
				2−2−2	2.573	2.560	**18.881**	-
				7−1	2.165	2.158	16.866	-
			RSD-S	2×7	**3.095**	**3.059**	18.650	-
				7×2	2.275	2.267	17.554	-
		21	SD	21	2.947	2.846	11.708	-
			SpecTr	3×7	3.027	2.992	18.297	-
				7×3	2.556	2.543	18.009	-
			RSD-C	3−1−1−1−1−1−1	2.877	2.843	17.909	-
				3−2−2	2.604	2.591	19.223	-
				7−1−1	2.442	2.430	18.094	-
			RSD-S	3×7	**3.229**	**3.191**	18.924	-
				7×3	2.668	2.655	**19.251**	-
		30	SD	30	2.956	2.813	9.762	-
			SpecTr	5×6	3.054	3.023	19.007	-
				6×5	2.958	2.933	19.088	-
			RSD-C	2−2−2−2	2.830	2.810	19.703	-
				5−1−1−1−1−1	2.831	2.803	18.427	-
				6−1−1−1−1	2.748	2.725	18.699	-
			RSD-S	5×6	**3.234**	**3.201**	19.812	-
				6×5	3.127	3.101	**20.308**	-

Table 43: We summarize experiment results with OPT 13B target and 125M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-13B	XSum	1	AR	-	1.000	1.000	38.711	0.127
		6	SD	6	**2.133**	**2.015**	23.065	0.126
			SpecTr	2×3	1.962	1.906	27.405	0.124
				3×2	1.833	1.798	30.186	0.127
			RSD-C	2−1−1	1.988	1.931	28.655	0.129
				2−2	1.909	1.872	31.400	0.124
				3−1	1.854	1.818	30.365	0.127
			RSD-S	2×3	2.035	1.977	29.294	0.128
				3×2	1.930	1.893	**32.040**	0.124
		10	SD	10	2.205	2.009	16.652	0.128
			SpecTr	2×5	2.132	2.033	23.528	0.127
				5×2	1.829	1.794	29.909	0.124
			RSD-C	2−1−1−1−1	2.126	2.027	23.528	0.126
				2−2−1	2.043	1.985	28.339	0.126
				5−1	1.878	1.843	30.072	0.128
			RSD-S	2×5	**2.269**	**2.163**	24.880	0.121
				5×2	1.969	1.931	**31.013**	0.126
		14	SD	14	2.221	1.954	13.362	0.126
			SpecTr	2×7	2.309	2.162	20.931	0.123
				7×2	1.894	1.858	30.480	0.127
			RSD-C	2−1−1−1−1−1−1	2.164	2.026	20.166	0.122
				2−2−2	2.126	2.065	28.965	0.127
				7−1	1.892	1.856	30.249	0.125
			RSD-S	2×7	**2.329**	**2.180**	21.482	0.127
				7×2	2.064	2.024	**32.678**	0.127
		21	SD	21	2.262	1.878	9.417	0.128
			SpecTr	3×7	2.223	2.081	19.577	0.128
				7×3	2.030	1.973	28.103	0.127
			RSD-C	3−1−1−1−1−1−1	2.207	2.066	20.300	0.130
				3−2−2	2.154	2.093	29.846	0.126
				7−1−1	2.085	2.025	29.022	0.126
			RSD-S	3×7	**2.483**	**2.324**	22.530	0.128
				7×3	2.258	2.194	**30.439**	0.124
		30	SD	30	2.282	1.766	7.255	0.125
			SpecTr	5×6	2.260	2.135	22.044	0.126
				6×5	2.264	2.159	24.471	0.128
			RSD-C	2−2−2−2	2.248	2.164	**26.729**	0.127
				5−1−1−1−1−1	2.326	2.197	22.974	0.126
				6−1−1−1−1	2.260	2.155	24.486	0.129
			RSD-S	5×6	2.446	2.311	22.820	0.121
				6×5	**2.503**	**2.387**	25.887	0.124

Table 44: We summarize experiment results with OPT 13B target and 125M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-13B	WMT	1	AR	-	1.000	1.000	37.069	0.318
		6	SD	6	1.489	1.406	16.475	0.320
			SpecTr	2×3	1.512	1.469	21.669	0.318
				3×2	1.493	1.464	25.049	0.323
			RSD-C	2−1−1	1.557	1.513	22.541	0.317
				2−2	1.576	1.546	26.527	0.320
				3−1	1.592	1.561	26.282	0.320
			RSD-S	2×3	1.601	1.555	23.367	0.316
				3×2	**1.630**	**1.598**	**26.978**	0.320
		10	SD	10	1.494	1.362	11.782	0.320
			SpecTr	2×5	1.544	1.472	17.808	0.318
				5×2	1.554	1.524	26.354	0.321
			RSD-C	2−1−1−1−1	1.571	1.498	18.300	0.315
				2−2−1	1.614	1.568	23.393	0.321
				5−1	1.617	1.586	26.207	0.315
			RSD-S	2×5	1.629	1.553	18.628	0.318
				5×2	**1.713**	**1.680**	**27.673**	0.319
		14	SD	14	1.493	1.313	8.985	0.317
			SpecTr	2×7	1.551	1.452	14.843	0.321
				7×2	1.551	1.521	25.560	0.316
			RSD-C	2−1−1−1−1−1−1	1.584	1.483	14.929	0.314
				2−2−2	1.658	1.611	23.786	0.317
				7−1	1.644	1.613	26.398	0.320
			RSD-S	2×7	1.637	1.533	15.361	0.319
				7×2	**1.764**	**1.730**	**27.600**	0.318
		21	SD	21	1.491	1.238	6.465	0.315
			SpecTr	3×7	1.586	1.485	14.771	0.319
				7×3	1.599	1.553	22.628	0.317
			RSD-C	3−1−1−1−1−1−1	1.629	1.525	15.355	0.320
				3−2−2	1.732	1.683	24.226	0.317
				7−1−1	1.695	1.647	23.977	0.319
			RSD-S	3×7	1.730	1.619	16.081	0.318
				7×3	**1.839**	**1.787**	**24.796**	0.317
		30	SD	30	1.491	1.154	4.812	0.316
			SpecTr	5×6	1.616	1.527	16.789	0.320
				6×5	1.668	1.590	18.352	0.321
			RSD-C	2−2−2−2	1.682	1.619	**21.212**	0.319
				5−1−1−1−1−1	1.686	1.593	16.849	0.314
				6−1−1−1−1	1.751	1.669	19.230	0.316
			RSD-S	5×6	1.838	1.736	18.155	0.314
				6×5	**1.860**	**1.774**	19.945	0.316

Table 45: We summarize experiment results with OPT 30B target and 125M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-30B	XSum	1	AR	-	1.000	1.000	20.633	0.126
		6	SD	6	**2.323**	**2.266**	19.542	0.124
			SpecTr	2×3	2.199	2.172	23.748	0.122
				3×2	1.944	1.928	23.499	0.127
			RSD-C	2−1−1	2.214	2.186	23.832	0.122
				2−2	1.995	1.978	23.985	0.121
				3−1	1.944	1.928	23.348	0.121
			RSD-S	2×3	2.196	2.168	23.299	0.124
				3×2	2.032	2.015	**24.170**	0.123
		10	SD	10	**2.544**	**2.442**	16.464	0.123
			SpecTr	2×5	2.361	2.313	20.368	0.121
				5×2	1.962	1.946	23.209	0.127
			RSD-C	2−1−1−1−1	2.314	2.266	20.594	0.123
				2−2−1	2.234	2.206	23.613	0.127
				5−1	2.011	1.994	23.874	0.122
			RSD-S	2×5	2.468	2.418	21.438	0.126
				5×2	2.106	2.089	**24.545**	0.117
		14	SD	14	2.556	2.415	13.274	0.126
			SpecTr	2×7	2.527	2.455	18.833	0.121
				7×2	1.985	1.968	23.429	0.122
			RSD-C	2−1−1−1−1−1−1	2.577	2.504	19.684	0.124
				2−2−2	2.236	2.209	23.572	0.121
				7−1	2.011	1.994	23.608	0.121
			RSD-S	2×7	**2.665**	**2.589**	19.724	0.120
				7×2	2.107	2.089	**24.230**	0.126
		21	SD	21	2.647	2.433	10.217	0.124
			SpecTr	3×7	2.617	2.543	19.371	0.127
				7×3	2.170	2.143	22.653	0.118
			RSD-C	3−1−1−1−1−1−1	2.640	2.565	19.956	0.117
				3−2−2	2.350	2.321	**24.503**	0.123
				7−1−1	2.206	2.179	22.493	0.120
			RSD-S	3×7	**2.778**	**2.699**	20.585	0.121
				7×3	2.391	2.362	24.425	0.128
		30	SD	30	2.677	2.379	7.703	0.126
			SpecTr	5×6	2.583	2.519	20.213	0.125
				6×5	2.398	2.349	20.342	0.125
			RSD-C	2−2−2−2	2.509	2.468	**23.328**	0.120
				5−1−1−1−1−1	2.551	2.488	20.177	0.121
				6−1−1−1−1	2.335	2.287	19.893	0.124
			RSD-S	5×6	2.686	2.620	20.690	0.123
				6×5	**2.746**	**2.690**	22.804	0.121

Table 46: We summarize experiment results with OPT 30B target and 125M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-30B	WMT	1	AR	-	1.000	1.000	19.162	0.347
		6	SD	6	1.471	1.435	12.745	0.341
			SpecTr	2×3	1.496	1.477	16.309	0.345
				3×2	1.480	1.468	17.775	0.345
			RSD-C	2−1−1	1.535	1.516	16.667	0.340
				2−2	1.563	1.550	18.667	0.344
				3−1	1.546	1.533	18.126	0.342
			RSD-S	2×3	1.583	1.563	16.954	0.344
				3×2	**1.609**	**1.596**	**18.783**	0.344
		10	SD	10	1.475	1.416	9.537	0.346
			SpecTr	2×5	1.519	1.488	13.829	0.345
				5×2	1.514	1.502	17.872	0.341
			RSD-C	2−1−1−1−1	1.556	1.524	13.865	0.343
				2−2−1	1.633	1.612	17.388	0.338
				5−1	1.610	1.597	18.691	0.344
			RSD-S	2×5	1.613	1.580	14.255	0.344
				5×2	**1.694**	**1.680**	**19.514**	0.341
		14	SD	14	1.472	1.390	7.912	0.344
			SpecTr	2×7	1.527	1.483	11.671	0.345
				7×2	1.525	1.512	17.959	0.342
			RSD-C	2−1−1−1−1−1−1	1.562	1.518	12.081	0.342
				2−2−2	1.623	1.603	17.090	0.343
				7−1	1.655	1.641	18.792	0.341
			RSD-S	2×7	1.620	1.574	12.481	0.342
				7×2	**1.737**	**1.722**	**19.383**	0.340
		21	SD	21	1.473	1.355	5.851	0.340
			SpecTr	3×7	1.559	1.514	11.990	0.342
				7×3	1.578	1.559	16.725	0.344
			RSD-C	3−1−1−1−1−1−1	1.609	1.564	12.281	0.340
				3−2−2	1.709	1.688	17.662	0.347
				7−1−1	1.669	1.649	17.303	0.343
			RSD-S	3×7	1.706	1.658	12.704	0.344
				7×3	**1.813**	**1.790**	**18.205**	0.344
		30	SD	30	1.478	1.313	4.415	0.346
			SpecTr	5×6	1.598	1.559	12.840	0.341
				6×5	1.599	1.566	14.059	0.343
			RSD-C	2−2−2−2	1.658	1.631	**15.863**	0.339
				5−1−1−1−1−1	1.669	1.628	13.523	0.345
				6−1−1−1−1	1.683	1.649	14.331	0.348
			RSD-S	5×6	**1.854**	**1.808**	14.459	0.341
				6×5	1.837	1.800	15.534	0.338

Table 47: We summarize experiment results with OPT 66B target and 125M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-66B	XSum	1	AR	-	1.000	1.000	9.454	0.125
		6	SD	6	**2.810**	**2.778**	14.393	0.123
			SpecTr	2×3	2.394	2.381	14.287	0.119
				3×2	2.140	2.132	14.865	0.121
			RSD-C	2−1−1	2.432	2.418	**15.667**	0.119
				2−2	2.122	2.114	14.806	0.122
				3−1	2.139	2.131	13.853	0.123
			RSD-S	2×3	2.383	2.370	14.758	0.125
				3×2	2.218	2.210	15.415	0.121
		10	SD	10	**3.027**	**2.970**	12.300	0.122
			SpecTr	2×5	2.901	2.874	14.880	0.125
				5×2	2.142	2.134	14.721	0.124
			RSD-C	2−1−1−1−1	2.651	2.626	14.175	0.125
				2−2−1	2.436	2.422	15.514	0.126
				5−1	2.186	2.178	14.726	0.125
			RSD-S	2×5	2.891	2.864	**15.652**	0.129
				5×2	2.256	2.248	15.329	0.122
		14	SD	14	3.030	2.951	10.207	0.122
			SpecTr	2×7	3.155	3.114	14.842	0.123
				7×2	2.158	2.150	14.811	0.124
			RSD-C	2−1−1−1−1−1−1	2.964	2.925	13.944	0.120
				2−2−2	2.644	2.629	**16.681**	0.119
				7−1	2.189	2.181	13.765	0.122
			RSD-S	2×7	**3.244**	**3.202**	15.033	0.126
				7×2	2.265	2.257	15.450	0.124
		21	SD	21	3.272	3.146	8.545	0.121
			SpecTr	3×7	3.099	3.059	14.581	0.124
				7×3	2.393	2.379	15.093	0.121
			RSD-C	3−1−1−1−1−1−1	3.028	2.988	14.228	0.126
				3−2−2	2.432	2.418	13.788	0.122
				7−1−1	2.464	2.450	14.192	0.126
			RSD-S	3×7	**3.382**	**3.338**	15.426	0.122
				7×3	2.527	2.513	**15.569**	0.126
		30	SD	30	**3.345**	3.164	6.731	0.123
			SpecTr	5×6	3.111	3.076	15.069	0.124
				6×5	2.852	2.825	15.102	0.121
			RSD-C	2−2−2−2	2.900	2.878	15.344	0.129
				5−1−1−1−1−1	3.185	3.149	15.784	0.120
				6−1−1−1−1	2.990	2.962	15.892	0.123
			RSD-S	5×6	3.283	**3.246**	**16.125**	0.122
				6×5	2.920	2.893	14.856	0.122

Table 48: We summarize experiment results with OPT 66B target and 125M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-125M-66B	WMT	1	AR	-	1.000	1.000	9.306	0.359
		6	SD	6	1.468	1.452	7.673	0.359
			SpecTr	2×3	1.502	1.493	9.690	0.360
				3×2	1.486	1.481	10.265	0.356
			RSD-C	2−1−1	1.542	1.533	9.687	0.356
				2−2	1.570	1.564	10.894	0.361
				3−1	1.557	1.551	10.544	0.360
			RSD-S	2×3	1.589	1.580	10.212	0.362
				3×2	**1.619**	**1.613**	**10.951**	0.361
		10	SD	10	1.476	1.449	6.064	0.357
			SpecTr	2×5	1.527	1.512	8.530	0.359
				5×2	1.520	1.514	10.550	0.359
			RSD-C	2−1−1−1−1	1.570	1.555	8.506	0.359
				2−2−1	1.595	1.586	10.189	0.357
				5−1	1.615	1.609	11.081	0.355
			RSD-S	2×5	1.619	1.604	8.784	0.358
				5×2	**1.697**	**1.691**	**11.539**	0.361
		14	SD	14	1.483	1.445	5.163	0.357
			SpecTr	2×7	1.539	1.518	7.344	0.354
				7×2	1.556	1.550	10.707	0.357
			RSD-C	2−1−1−1−1−1−1	1.572	1.551	7.527	0.363
				2−2−2	1.635	1.625	10.345	0.361
				7−1	1.641	1.635	11.190	0.356
			RSD-S	2×7	1.629	1.608	7.755	0.361
				7×2	**1.771**	**1.765**	**11.578**	0.357
		21	SD	21	1.473	1.416	3.936	0.359
			SpecTr	3×7	1.574	1.553	7.470	0.357
				7×3	1.588	1.579	10.017	0.361
			RSD-C	3−1−1−1−1−1−1	1.628	1.606	7.733	0.361
				3−2−2	1.717	1.707	10.731	0.358
				7−1−1	1.685	1.675	10.545	0.359
			RSD-S	3×7	1.716	1.693	8.120	0.359
				7×3	**1.830**	**1.820**	**11.197**	0.357
		30	SD	30	1.473	1.393	2.996	0.357
			SpecTr	5×6	1.603	1.584	8.073	0.360
				6×5	1.639	1.624	8.815	0.356
			RSD-C	2−2−2−2	1.669	1.656	**9.674**	0.360
				5−1−1−1−1−1	1.676	1.657	8.378	0.358
				6−1−1−1−1	1.692	1.676	8.944	0.355
			RSD-S	5×6	1.817	1.797	8.953	0.360
				6×5	**1.846**	**1.829**	9.663	0.359

Table 49: We summarize experiment results with OPT 13B target and 350M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-13B	XSum	1	AR	-	1.000	1.000	37.931	0.130
		6	SD	6	1.892	1.638	13.451	0.128
			SpecTr	2×3	1.874	1.739	20.428	0.131
				3×2	1.727	1.642	23.011	0.132
			RSD-C	2−1−1	1.844	1.711	19.973	0.126
				2−2	1.793	1.705	**24.072**	0.125
				3−1	1.739	1.654	23.247	0.125
			RSD-S	2×3	**1.926**	**1.787**	20.940	0.126
				3×2	1.808	1.720	23.488	0.129
		10	SD	10	**2.049**	1.629	9.717	0.127
			SpecTr	2×5	1.992	1.765	15.340	0.130
				5×2	1.792	1.705	23.852	0.121
			RSD-C	2−1−1−1−1	1.929	1.709	15.067	0.130
				2−2−1	1.861	1.727	19.849	0.123
				5−1	1.808	1.719	**24.163**	0.126
			RSD-S	2×5	2.046	**1.812**	16.124	0.124
				5×2	1.837	1.747	24.070	0.125
		14	SD	14	1.968	1.447	6.942	0.125
			SpecTr	2×7	1.990	1.686	12.126	0.129
				7×2	1.747	1.661	22.925	0.130
			RSD-C	2−1−1−1−1−1−1	1.983	1.680	12.010	0.129
				2−2−2	1.965	1.824	21.138	0.125
				7−1	1.809	1.721	23.633	0.128
			RSD-S	2×7	**2.174**	**1.842**	13.189	0.124
				7×2	1.873	1.781	**24.198**	0.130
		21	SD	21	2.090	1.356	5.206	0.125
			SpecTr	3×7	2.139	1.812	13.218	0.125
				7×3	1.893	1.757	20.078	0.130
			RSD-C	3−1−1−1−1−1−1	2.080	1.762	12.446	0.132
				3−2−2	1.945	1.806	20.561	0.126
				7−1−1	1.874	1.739	19.448	0.124
			RSD-S	3×7	**2.243**	1.900	13.525	0.128
				7×3	2.083	**1.934**	**21.720**	0.128
		30	SD	30	2.098	1.183	3.760	0.125
			SpecTr	5×6	2.106	1.824	14.311	0.125
				6×5	2.044	1.811	15.561	0.127
			RSD-C	2−2−2−2	1.975	1.791	**17.488**	0.125
				5−1−1−1−1−1	2.068	1.791	13.742	0.127
				6−1−1−1−1	2.094	1.855	16.298	0.125
			RSD-S	5×6	**2.283**	1.977	15.204	0.128
				6×5	2.274	**2.015**	17.290	0.127

Table 50: We summarize experiment results with OPT 13B target and 350M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-13B	WMT	1	AR	-	1.000	1.000	41.479	0.316
		6	SD	6	1.308	1.133	9.562	0.321
			SpecTr	2×3	1.331	1.236	14.912	0.322
				3×2	1.327	1.262	18.494	0.319
			RSD-C	2−1−1	1.356	1.259	15.159	0.318
				2−2	1.397	1.329	19.025	0.318
				3−1	1.379	1.311	18.864	0.320
			RSD-S	2×3	1.379	1.280	15.313	0.315
				3×2	**1.399**	**1.330**	**19.321**	0.320
		10	SD	10	1.313	1.044	6.306	0.317
			SpecTr	2×5	1.340	1.187	10.872	0.321
				5×2	1.345	1.279	18.768	0.315
			RSD-C	2−1−1−1−1	1.367	1.211	11.184	0.322
				2−2−1	1.384	1.285	15.366	0.322
				5−1	1.411	1.342	18.759	0.320
			RSD-S	2×5	1.391	1.232	11.165	0.318
				5×2	**1.447**	**1.376**	**19.953**	0.319
		14	SD	14	1.308	0.961	4.752	0.322
			SpecTr	2×7	1.345	1.140	8.473	0.319
				7×2	1.356	1.289	18.334	0.321
			RSD-C	2−1−1−1−1−1−1	1.414	1.198	8.830	0.314
				2−2−2	1.395	1.295	15.460	0.315
				7−1	1.429	1.359	19.081	0.314
			RSD-S	2×7	1.397	1.184	8.920	0.319
				7×2	**1.480**	**1.408**	**19.201**	0.319
		21	SD	21	1.311	0.851	3.315	0.321
			SpecTr	3×7	1.363	1.155	8.567	0.318
				7×3	1.415	1.314	15.697	0.320
			RSD-C	3−1−1−1−1−1−1	1.398	1.184	8.514	0.323
				3−2−2	1.442	1.339	15.717	0.319
				7−1−1	1.480	1.374	16.062	0.314
			RSD-S	3×7	1.439	1.219	8.938	0.322
				7×3	**1.509**	**1.401**	**16.265**	0.324
		30	SD	30	1.310	0.739	2.363	0.319
			SpecTr	5×6	1.426	1.235	9.794	0.322
				6×5	1.390	1.231	11.136	0.318
			RSD-C	2−2−2−2	1.411	1.280	**13.218**	0.320
				5−1−1−1−1−1	1.470	1.273	10.229	0.313
				6−1−1−1−1	1.489	1.319	11.662	0.320
			RSD-S	5×6	1.493	1.293	10.522	0.319
				6×5	**1.516**	**1.343**	11.778	0.316

Table 51: We summarize experiment results with OPT 30B target and 350M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-30B	XSum	1	AR	-	1.000	1.000	20.127	0.125
		6	SD	6	**2.220**	2.082	13.680	0.123
			SpecTr	2×3	1.999	1.934	16.896	0.126
				3×2	1.872	1.831	18.900	0.124
			RSD-C	2−1−1	2.104	2.037	18.212	0.124
				2−2	1.952	1.910	19.635	0.122
				3−1	1.872	1.831	18.623	0.124
			RSD-S	2×3	2.171	**2.101**	18.713	0.122
				3×2	1.972	1.929	**19.967**	0.122
		10	SD	10	2.307	2.077	9.925	0.121
			SpecTr	2×5	2.220	2.104	14.424	0.124
				5×2	1.893	1.852	18.996	0.122
			RSD-C	2−1−1−1−1	2.313	2.192	14.995	0.122
				2−2−1	2.190	2.120	18.528	0.121
				5−1	1.950	1.908	19.789	0.124
			RSD-S	2×5	**2.399**	**2.273**	15.656	0.121
				5×2	2.035	1.991	**20.671**	0.125
		14	SD	14	2.345	2.031	7.754	0.126
			SpecTr	2×7	2.387	2.215	12.547	0.125
				7×2	1.913	1.872	19.244	0.124
			RSD-C	2−1−1−1−1−1−1	2.311	2.145	12.336	0.126
				2−2−2	2.082	2.015	17.433	0.124
				7−1	1.974	1.931	19.812	0.129
			RSD-S	2×7	**2.416**	**2.243**	12.801	0.121
				7×2	2.046	2.002	**20.472**	0.120
		21	SD	21	2.414	1.960	5.608	0.125
			SpecTr	3×7	2.505	2.325	13.243	0.123
				7×3	2.134	2.065	17.884	0.124
			RSD-C	3−1−1−1−1−1−1	**2.546**	**2.364**	13.544	0.125
				3−2−2	2.193	2.123	18.659	0.122
				7−1−1	2.137	2.068	18.110	0.123
			RSD-S	3×7	2.437	2.262	12.969	0.123
				7×3	2.251	2.179	**18.787**	0.121
		30	SD	30	2.496	1.875	4.309	0.118
			SpecTr	5×6	2.500	2.345	14.386	0.121
				6×5	2.399	2.274	15.473	0.124
			RSD-C	2−2−2−2	2.292	2.195	**16.656**	0.123
				5−1−1−1−1−1	2.275	2.134	13.027	0.128
				6−1−1−1−1	2.318	2.197	14.832	0.122
			RSD-S	5×6	**2.571**	**2.411**	14.623	0.123
				6×5	2.420	2.294	15.142	0.125

Table 52: We summarize experiment results with OPT 30B target and 350M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-30B	WMT	1	AR	-	1.000	1.000	20.127	0.341
		6	SD	6	1.307	1.226	8.167	0.338
			SpecTr	2×3	1.401	1.356	12.387	0.340
				3×2	1.324	1.295	13.946	0.342
			RSD-C	2−1−1	1.363	1.319	11.974	0.341
				2−2	1.378	1.348	14.204	0.350
				3−1	1.400	1.370	**14.558**	0.340
			RSD-S	2×3	1.382	1.338	12.350	0.342
				3×2	**1.407**	**1.376**	14.464	0.345
		10	SD	10	1.313	1.183	5.757	0.338
			SpecTr	2×5	1.340	1.270	9.075	0.341
				5×2	1.376	1.346	14.347	0.344
			RSD-C	2−1−1−1−1	1.372	1.301	9.373	0.346
				2−2−1	1.425	1.380	12.533	0.344
				5−1	1.417	1.386	14.496	0.344
			RSD-S	2×5	1.390	1.317	9.241	0.345
				5×2	**1.480**	**1.448**	**14.966**	0.346
		14	SD	14	1.305	1.130	4.367	0.344
			SpecTr	2×7	1.387	1.287	7.614	0.341
				7×2	1.355	1.326	13.878	0.343
			RSD-C	2−1−1−1−1−1−1	1.369	1.271	7.458	0.343
				2−2−2	1.403	1.358	12.124	0.349
				7−1	1.437	1.406	14.461	0.342
			RSD-S	2×7	1.393	1.293	7.487	0.341
				7×2	**1.489**	**1.456**	**14.794**	0.346
		21	SD	21	1.313	1.066	3.213	0.345
			SpecTr	3×7	1.359	1.262	7.408	0.343
				7×3	1.387	1.342	12.095	0.344
			RSD-C	3−1−1−1−1−1−1	1.406	1.305	7.805	0.343
				3−2−2	1.444	1.398	12.481	0.344
				7−1−1	1.456	1.409	12.513	0.343
			RSD-S	3×7	1.446	1.342	7.829	0.341
				7×3	**1.517**	**1.468**	**12.755**	0.342
		30	SD	30	1.311	0.984	2.273	0.346
			SpecTr	5×6	1.428	1.339	8.510	0.344
				6×5	1.388	1.315	9.105	0.345
			RSD-C	2−2−2−2	1.413	1.353	**10.584**	0.344
				5−1−1−1−1−1	1.484	1.392	8.946	0.345
				6−1−1−1−1	1.456	1.380	9.474	0.341
			RSD-S	5×6	1.500	1.407	8.849	0.343
				6×5	**1.517**	**1.438**	9.978	0.342

Table 53: We summarize experiment results with OPT 66B target and 350M draft for the XSum task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-66B	XSum	1	AR	-	1.000	1.000	9.537	0.123
		6	SD	6	**2.512**	**2.438**	9.748	0.118
			SpecTr	2×3	2.228	2.195	11.842	0.126
				3×2	1.932	1.913	11.880	0.124
			RSD-C	2−1−1	2.217	2.184	11.746	0.125
				2−2	2.020	1.999	11.982	0.125
				3−1	2.038	2.018	11.603	0.123
			RSD-S	2×3	2.291	2.257	**12.140**	0.126
				3×2	2.070	2.049	11.694	0.126
		10	SD	10	**2.704**	**2.574**	7.749	0.122
			SpecTr	2×5	2.452	2.392	10.203	0.125
				5×2	2.005	1.985	12.062	0.121
			RSD-C	2−1−1−1−1	2.627	2.563	10.912	0.123
				2−2−1	2.194	2.161	10.804	0.122
				5−1	2.001	1.981	11.154	0.121
			RSD-S	2×5	2.583	2.520	10.279	0.125
				5×2	2.098	2.077	**12.364**	0.123
		14	SD	14	2.616	2.443	5.912	0.127
			SpecTr	2×7	2.734	2.640	9.562	0.121
				7×2	2.045	2.025	12.290	0.121
			RSD-C	2−1−1−1−1−1−1	**2.865**	**2.768**	10.029	0.122
				2−2−2	2.325	2.290	11.967	0.126
				7−1	2.022	2.002	12.235	0.123
			RSD-S	2×7	2.609	2.520	9.184	0.122
				7×2	2.160	2.139	**12.650**	0.125
		21	SD	21	**2.938**	2.656	4.894	0.124
			SpecTr	3×7	2.580	2.492	8.900	0.124
				7×3	2.375	2.340	12.345	0.121
			RSD-C	3−1−1−1−1−1−1	2.770	**2.675**	9.639	0.116
				3−2−2	2.364	2.328	12.068	0.128
				7−1−1	2.256	2.222	11.705	0.123
			RSD-S	3×7	2.627	2.538	9.164	0.125
				7×3	2.527	2.489	**12.944**	0.123
		30	SD	30	**3.185**	2.767	3.802	0.126
			SpecTr	5×6	2.828	2.745	10.633	0.120
				6×5	2.652	2.586	10.857	0.123
			RSD-C	2−2−2−2	2.544	2.494	10.990	0.127
				5−1−1−1−1−1	2.742	2.662	10.176	0.117
				6−1−1−1−1	2.587	2.523	10.119	0.124
			RSD-S	5×6	2.723	2.643	9.942	0.124
				6×5	2.937	**2.865**	**11.984**	0.119

Table 54: We summarize experiment results with OPT 66B target and 350M draft for the WMT task with various target computational budgets. Target Complexity (Comp.) means the number of tokens parallelly evaluated at the target model. For decoders (Dec.), we consider Auto-Regressive sampling (AR), Speculative Decoding (SD), SpecTr, Recursive Speculative Decoding with Constant branching factors (RSD-C), Recursive Speculative Decoding with Stochastic Beam Search (RSD-S). The contents in decoder specification (Spec.) have different meanings for each decoder; $K \times L$ means the number K of draft paths and draft length L for SpecTr; it describes constant branching factors for each level of the tree (from root to leaf) for RSD-C; $K \times L$ means the beamwidth K and draft length L for RSD-S. Block efficiency (Eff.), Memory Bound Speed Up (MBSU), Token Rate (TR), and Accuracy (Acc.) are given for each algorithm. For each group of rows having the same complexity, we highlight the best values for all columns except accuracy.

Model	Task	Comp.	Dec.	Spec.	Eff.	MBSU	TR	Acc.
OPT-350M-66B	WMT	1	AR	-	1.000	1.000	9.422	0.355
		6	SD	6	1.297	1.259	5.151	0.359
			SpecTr	2×3	1.316	1.296	7.060	0.363
				3×2	1.313	1.300	8.007	0.358
			RSD-C	2−1−1	1.348	1.328	7.230	0.357
				2−2	1.353	1.339	8.227	0.358
				3−1	1.358	1.344	8.403	0.358
			RSD-S	2×3	1.359	1.339	7.294	0.362
				3×2	**1.390**	**1.376**	**8.621**	0.361
		10	SD	10	1.298	1.236	3.737	0.357
			SpecTr	2×5	1.327	1.295	5.608	0.358
				5×2	1.329	1.315	8.155	0.358
			RSD-C	2−1−1−1−1	1.355	1.322	5.700	0.353
				2−2−1	1.374	1.353	7.336	0.361
				5−1	1.399	1.385	8.365	0.357
			RSD-S	2×5	1.373	1.339	5.778	0.355
				5×2	**1.434**	**1.420**	**8.757**	0.359
		14	SD	14	1.297	1.211	2.974	0.360
			SpecTr	2×7	1.324	1.278	4.695	0.360
				7×2	1.365	1.351	8.347	0.355
			RSD-C	2−1−1−1−1−1−1	1.354	1.308	4.761	0.360
				2−2−2	1.384	1.364	7.417	0.357
				7−1	1.416	1.402	8.420	0.356
			RSD-S	2×7	1.381	1.334	4.904	0.357
				7×2	**1.464**	**1.450**	**8.682**	0.360
		21	SD	21	1.296	1.172	2.122	0.360
			SpecTr	3×7	1.343	1.297	4.702	0.355
				7×3	1.367	1.346	7.242	0.359
			RSD-C	3−1−1−1−1−1−1	1.389	1.341	4.840	0.362
				3−2−2	1.425	1.404	7.545	0.358
				7−1−1	1.435	1.413	7.484	0.358
			RSD-S	3×7	1.426	1.378	4.991	0.357
				7×3	**1.490**	**1.467**	**7.702**	0.360
		30	SD	30	1.293	1.123	1.577	0.363
			SpecTr	5×6	1.362	1.322	5.169	0.357
				6×5	1.373	1.340	5.756	0.360
			RSD-C	2−2−2−2	1.396	1.369	**6.424**	0.363
				5−1−1−1−1−1	1.423	1.381	5.402	0.353
				6−1−1−1−1	1.431	1.396	5.981	0.359
			RSD-S	5×6	1.477	1.434	5.591	0.360
				6×5	**1.489**	**1.453**	6.230	0.360