Multi-Candidate Speculative Decoding

Anonymous ACL submission

Abstract

Large language models have shown impressive capabilities across a variety of NLP tasks, yet their generating text autoregressively is time-consuming. One way to speed them up is speculative decoding, which generates candidate segments (a sequence of tokens) from a fast draft model that is then verified in parallel by the target model. However, the acceptance rate of candidate tokens from the draft model can be affected by several factors, such as the model, the dataset, and the decoding setup. This paper proposes to sample multiple candidates from a draft model and then organise them in batches for verification. We design algorithms for efficient multi-candidate verification while maintaining the distribution of the target model. Our approach shows significant improvements in acceptance rates across datasets, models, and decoding setups, consistently outperforming standard speculative decoding.[1]

1 Introduction

Recently developed large language models (LLMs), such as GPT series (Brown et al., 2020; Achiam et al., 2023) and LLaMA (Touvron et al., 2023a,b), have demonstrated remarkable capabilities in language understanding and generation, as well as generalizability across a wide variety of NLP tasks and open domains. This promotes the need to deploy LLM services. However, the extensive volume of parameters and computational overhead make LLMs run with significantly higher latency than smaller models. On the other hand, popular Transformer-based LLMs typically generate text in an autoregressive paradigm, which necessitates multiple iterations of the model for decoding a single piece of text, making things even worse.

To reduce the inference cost, a series of methods have have been developed (So et al., 2021; Shazeer, 2019; Kwon et al., 2023). Among them, *speculative decoding* (SD) (Leviathan et al., 2023; Chen et al., 2023) has been proved to be very effective in improving the end-to-end latency of large autoregressive models without compromising the quality of generation. SD employs an additional draft model, typically much smaller than the target model to be accelerated, to generate a sequence of tokens as candidate at low computational cost. These tokens are concurrently fed into the target model and conditionally accepted to preserve the output distribution of the target model.

The main purpose of SD is to minimize the invocations of the target model by accepting as many tokens as possible during the verification stage. Therefore, the acceleration performance crucially depends on the acceptance rate of candidate tokens by the target model, i.e., the agreement between the draft and target model's distributions under a given context. In general, models within the same suite generally exhibit strong agreement in their output distributions. However, our experiments reveal that the distributional discrepancies between the target and draft models become more pronounced when tackling complex tasks that involve longer prompts. On the other hand, it has become popular in the community to fine-tune LLMs using additional data to enhance their performance in specific aspects (Chiang et al., 2023; Taori et al., 2023; Iyer et al., 2022; Chung et al., 2022). It is important to note that fine-tuning can also introduce significant distributional discrepancies between the target and draft models, even if they are initially well-aligned.

The aim of this work is to improve the acceptance rate by sampling multiple candidate tokens at each position in the draft generation. These candidates can be organized in a batch for parallel verification on the target model. Although this approach is straightforward and intuitive, it encounters the challenge that SD cannot directly utilize multiple candidates to improve the acceptance rate

[1]We release codes, datasets, and model checkpoints at *Anonymous URL*.

while preserving the output distribution of the target model. To address it, we propose an algorithm for multi-candidate verification. Moreover, the multiple candidates sampled from the draft model have a probability of collision. Thus, we also introduce a more efficient version that avoids collisions by sampling candidates without replacement, and use Tree Attention to alleviate the memory-IO bound.

We evaluate our method using the LLaMA suite, including its fine-tuned version, Vicuna, with both argmax and standard sampling. Our method yields significant improvements in acceptance rates on the Alpaca and WMT datasets, consistently outperforming SD in walltime speed. We further validate our method's generalizability across models by extending it to the LLaMA2 and OPT suites. Notably, acceptance rates can often be improved by fine-tuning the draft model, and we show that our method can be superimposed on it.

2 Background: Speculative Decoding

The workflow of speculative decoding is shown in Fig. 1(a). Given contexts c, speculative decoding starts by invoking the draft model M_q to sample a draft sequence of tokens with a length of γ, denoted as $\tilde{x}_1, \cdots, \tilde{x}_\gamma$, where $\tilde{x}_i \sim q(x|\tilde{x}_1, \cdots, \tilde{x}_{i-1}, c)$. The draft tokens, along with the contexts, are then passed to the target model M_p to obtain their output distribution $p(x|\tilde{x}_1, \cdots, \tilde{x}_i, c)$ in parallel. Finally, the draft tokens is verified sequentially from $\tilde{x}_1$ to $\tilde{x}_\gamma$. To verify token $\tilde{x}_i$, a speculative sampling algorithm is employed to determine whether to accept $\tilde{x}_i$ or not, based on $q(x|\tilde{x}, \cdots, \tilde{x}_{i-1}, c)$ and $p(x|\tilde{x}, \cdots, \tilde{x}_{i-1}, c)$. Once a token is rejected, the next verification terminates and the algorithm returns a new token as the endpoint. If all tokens are accepted, there is an extra token sampled from $p(x|\tilde{x}_1, \cdots, \tilde{x}_\gamma, c)$ as the endpoint. Thus, the process generates a minimum of 1 and a maximum of $\gamma + 1$ accepted tokens.

Speculative Sampling. The significance of speculative sampling is that we cannot accept the guesses given by the draft model without restriction, otherwise we cannot preserve the same output distribution as the target model. A simple and straightforward idea is to first sample a token x from $p(x)$[2], accept $\tilde{x}$ if $x = \tilde{x}$, otherwise reject it and return x. However, this approach — what we call *naive sampling* — has a very inefficient acceptance rate of $\sum_{\tilde{x}} p(\tilde{x})q(\tilde{x})$. As a comparison, speculative sampling uses a novel algorithm that accepts $\tilde{x}$ with probability $\min(1, \frac{p(\tilde{x})}{q(\tilde{x})})$, leading to an overall acceptance rate of $\sum_{\tilde{x}} \min(p(\tilde{x}), q(\tilde{x}))$. If $\tilde{x}$ is rejected, then return a new token sampling from $p'(x) = \frac{\max(0, p(x) - q(x))}{\sum_x \max(0, p(x) - q(x))}$. It can be proven that speculative sampling can preserve the same output distribution as the target model (Leviathan et al., 2023; Chen et al., 2023) while possessing an upper bound on the acceptance rate of naive sampling (Appendix C).

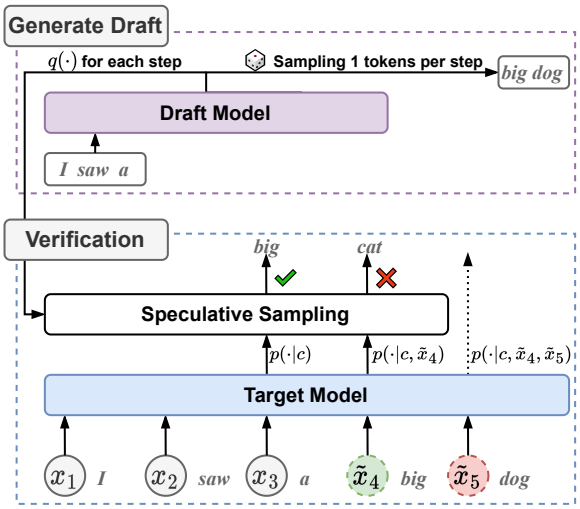
[2]We'll use $p(x)$ and $p(\tilde{x})$ to denote $p(x|\tilde{x}_1, \cdots, \tilde{x}_{i-1}, c)$ and $p(\tilde{x}_i|\tilde{x}_1, \cdots, \tilde{x}_{i-1}, c)$ respectively whenever the prefix is clear from the context, and similarly for $q(x)$ and $q(\tilde{x})$.

3 Multi-Candidate Speculative Decoding

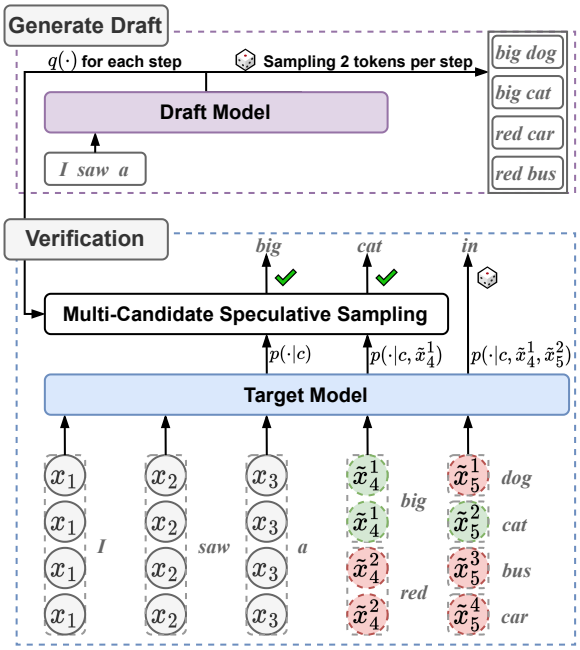
Behind the success of SD lies the effective utilization of parallel computing on computing devices: the latency of parallel scoring of short continuations is comparable to that of sampling a single token from the larger target model. Ideally, the length of draft tokens generated by the draft model (i.e., γ) can be increased all the way up to the upper limit of computing devices. However, there comes the diminishing marginal utility with an increase in γ rapidly, as the acceptance of a draft token for a given position depends not only on itself but also on the acceptance of all preceding tokens.

In short, there is a portion of potential computing resources that have not been fully utilized. Our work involves utilizing this portion of resources to perform parallel verification on another dimension (i.e., the batch dimension) to significantly improve the acceptance rate of draft tokens. This requires the draft model to sample more than one token at each step, eventually generating a batch of candidates. This batch of candidates is fed together into the target model to obtain the output distribution at each position, as shown in Fig. 1(b).

The verification process for multiple candidates is roughly the same as SD: starting from the first step of generation, input the output distributions q, p and candidate tokens $\tilde{x}^1, \cdots, \tilde{x}^k$ for the current step into the speculative sampling algorithm. If the algorithm accepts one of the k tokens, the candidate corresponding to that token is retained in the batch for the next step of verification. If the tokens are all rejected, the algorithm returns a new token as the endpoint, and the next verification procedure is aborted. Finally, if a candidate in the batch survives to the end, the endpoint token is sampled from the output distribution corresponding to this candidate. Taking a look at the process, it is



(a)



(b)

Figure 1: The procedure for standard SD (a) and MCSD (b).

similar to walking from the root of a tree to a leaf node, where each step chooses a path from one of the k branches or aborts early.

3.1 Multi-Candidate Speculative Sampling

Now the problem is that the original speculative sampling algorithm described in Section 2 cannot be directly used for verification of multiple candidates. This motivates us to design the speculative sampling algorithms for multi-candidate verification, as shown in Algorithm 1. The algorithm operates in the same manner as the standard SD to verify each candidate token. However, the distribution p is recalibrated based on q whenever a token fails verification. In Appendix A, we prove that the tokens output from Algorithm 1 follow the distribution $p(x)$, which ensures that the output obtained by our algorithm has the same distribution as the target model.

Algorithm 1: Multi-Candidate Speculative Sampling

Input: Distributions p, q; The candidate tokens $\tilde{x}^1, \cdots, \tilde{x}^k \sim q$.
Output: If accept, returns *True* and the accepted token, otherwise returns *False* and a new token as the endpoint.

for $i \leftarrow 1$ **to** k **do**
  $r \sim U(0,1)$
  if $r \leq \frac{p(\tilde{x}^i)}{q(\tilde{x}^i)}$ **then** ▷ Accept $\tilde{x}^i$
    return $(True, \tilde{x}^i)$
  else
    ▷ Normalize the residual $p(x)$
    $p(x) := \frac{\max(0, p(x) - q(x))}{\sum_x \max(0, p(x) - q(x))}$
end
$x_{\text{end}} \sim p(x)$
return $(False, x_{\text{end}})$

Algorithm 2: Multi-Candidate Speculative Sampling without Replacement

Input: Distributions p, q; The candidate tokens $\tilde{x}^1, \cdots, \tilde{x}^k$, where $\tilde{x}^i \sim \bar{q}_{i-1}(x)$.
Output: If accept, returns *True* and the accepted token, otherwise returns *False* and a new token as the endpoint.

for $i \leftarrow 1$ **to** k **do**
  $r \sim U(0,1)$
  if $r \leq \frac{p(\tilde{x}^i)}{q(\tilde{x}^i)}$ **then** ▷ Accept $\tilde{x}^i$
    return $(True, \tilde{x}^i)$
  else
    ▷ Normalize the residual $p(x)$ and $q(x)$
    $p(x) := \frac{\max(0, p(x) - q(x))}{\sum_x \max(0, p(x) - q(x))}$
    $q(\tilde{x}^i) \leftarrow 0$
    $q(x) := \frac{q(x)}{\sum_x q(x)}$
end
$x_{\text{end}} \sim p(x)$
return $(False, x_{\text{end}})$

3.2 Multi-Candidate Speculative Sampling without Replacement

Notice that the k tokens sampled by the draft model at each step are independently and identically distributed. As k increases, the possibility of collisions arises, meaning that there may be duplicates in $\tilde{x}^1, \cdots, \tilde{x}^k$. Even if a token is repeatedly sampled, there is no way to increase the probability of accepting it, because once it is rejected the first time, it has a probability of 0 in the residual distribution p, and thus will never be accepted again.

An intuitive way to prevent token collisions is to do sampling without replacement in draft model generation. Without loss of generality, assume that $\tilde{x}^1, \cdots, \tilde{x}^k$ are sampled without replacement from q sequentially, that is $\tilde{x}^i \sim \bar{q}_{i-1}(x)$, where

$$\bar{q}_0(x) = q(x),$$
$$\bar{q}_i(x) = \begin{cases} 0, & x \in \{\tilde{x}^1, \cdots, \tilde{x}^i\} \\ \frac{q(x)}{\sum_{x \neq \tilde{x}^1, \cdots, \tilde{x}^i} q(x)}, & \text{otherwise} \end{cases}. \quad (1)$$

We also propose the version without replacement as shown in Algorithm 2 and the proof that it preserves the distribution $p(x)$ in Appendix B.

3.3 Tree Attention

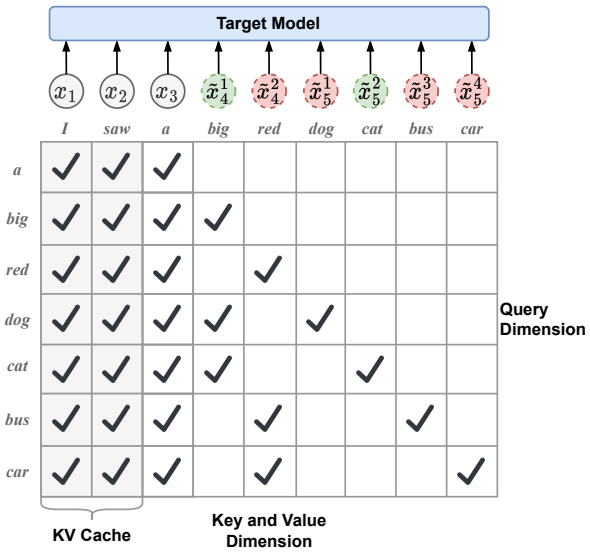


Figure 2: Processing multiple candidates in a single sequence concurrently based on Tree Attention, which contains an attention mask that exclusively permits each token to access its ancestors.

The generative Transformer (Vaswani et al., 2017), which serves as the backbone of LLMs, employs a left-to-right manner in text generation based on causal language modeling. To generate each new token, the attention mechanism requires accessing the keys and values of all preceding tokens. Due to the forward dependency, once a token is generated, the keys and values at that position remain unchanged in the following iterations, so it is very common to cache the keys and values of generated tokens for reuse. In multiple candidate generation and verification, these cached keys and values should be copied within the batch, for both the target and draft models. Although the copying incurs a slight overhead, the situation worsens as these keys and values need to be transferred to the computational unit at each layer of the model, even if they are numerically duplicated.

Here we employ Tree Attention (Miao et al., 2023; Cai et al., 2023) to mitigate the communication overhead resulting from replicated caches, which enables multiple candidates to share the caches of generated tokens. This time, all candidates are squeezed onto a single sequence in the order in which they were generated[3]. Then, a well-designed attention mask is applied to the sequence to prevent information contamination among candidates and preserve causal relationships between tokens, as shown in Fig. 2. With multiple candidate sequences arranged in a single sequence, the length of the sequence is slightly increased compared to the original. Accordingly, the total computational overhead is increased, but it is negligible compared to the communication overhead saved, since the contextual prefixes are generally much longer than the length of the candidates generated by the draft model. In our architecture, multiple candidates can be maximally squeezed into a single sequence without adding too much length, thanks to the k-ary tree formed by the candidate tokens, which allows a previously generated token to be shared by its descendants in the sequence.

4 Experiments

4.1 Experiment Settings

Models. Our evaluation is based on the publicly available LLM suite LLaMA (Touvron et al., 2023a), as well as its fine-tuned version Vicuna (Chiang et al., 2023), which is fine-tuned with instruction data to better perform dialogues and instructions. We select the 13B and 33B size versions as target models. Since there are no small models suitable for rapid draft generation included in the LLaMA suite, we employ LLaMA-68M and LLaMA-160M (Miao et al., 2023) as draft models, which are trained from scratch on the Wikipedia dataset and part of the C4 dataset (Raffel et al., 2020).

Datasets. We evaluate our approach on the conversational dataset Alpaca (Taori et al., 2023; Peng et al., 2023) and the translation dataset WMT EnDe (Bojar et al., 2014), which were used in previous works (Leviathan et al., 2023; Zhou et al., 2023) to evaluate decoding acceleration for LLM, and we observe that the performance of SD varies dramatically between the two datasets. For the

[3]The positional encoding of each token keeps the respective position in the previous sequence unchanged.

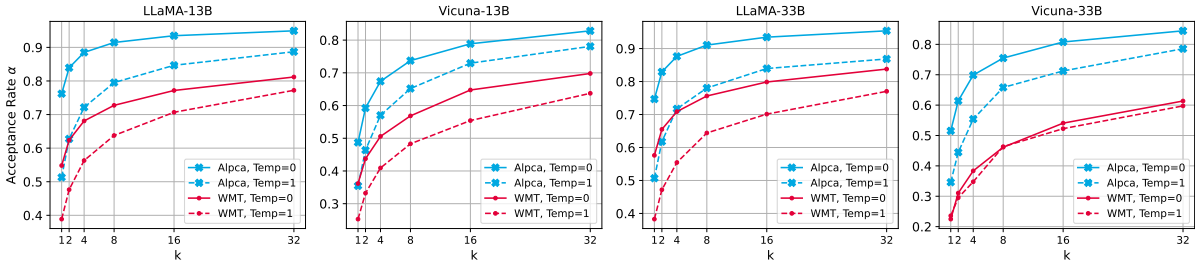


Figure 3: Acceptance rate (α) curves given different k using LLaMA-68M as draft model.

WMT dataset, the source text is embedded in an instruction template to prompt the model to perform the translation task.

Efficiency Measures. We use the **acceptance rate** α to evaluate the probability that a candidate token is accepted at each step, which basically indicates the distributional consistency between the draft model and the target model.

Since the draft model generates candidate segments of γ tokens at a time for verification, the **block efficiency** τ is commonly used as the expected number of generated tokens per block (Leviathan et al., 2023). Note that in the worst case, $\tau = 1$, since the algorithm at least returns a token as the endpoint; if all candidate tokens are accepted, the target model appends an extra token at the end, see Fig. 1(b), and so $\tau = \gamma + 1$.

In a real-world deployment, calling the draft model and executing our algorithm incurs additional overheads, so we report average wall clock speedups besides block efficiency.

Platform. The experiments were conducted on a single node, of which is equipped with four NVIDIA RTX A6000 48GB GPUs. All systems serves in half precision. The 13B versions are deployed on one GPU and the 33B versions are deployed across two GPUs. The draft models are deployed along with the target models and does not occupy additional GPUs.

Inference Settings. Since LLaMA base models are unlikely to stop generating naturally, we limit the generation length to a maximum of 128 new tokens. Our experiments involve two popular sampling setups, argmax sampling (temp=0) and standard sampling (temp=1). For other sampling methods, they can all easily be cast into standard sampling from an adjusted probability distribution.

4.2 Acceptance Rate Improvement

We begin by looking at the impact of different factors, such as the dataset, supervised fine-tuning,

Draft model	Target model	Temp	Alpaca	WMT
LLaMA-68M	LLaMA-13B	0	0.76	0.55
	Vicuna-13B		0.49	0.36
	LLaMA-33B		0.75	0.58
	Vicuna-33B		0.52	0.22
	LLaMA-13B	1	0.51	0.39
	Vicuna-13B		0.36	0.25
	LLaMA-33B		0.51	0.38
	Vicuna-33B		0.35	0.24
LLaMA-160M	LLaMA-13B	0	0.80	0.59
	Vicuna-13B		0.54	0.39
	LLaMA-33B		0.78	0.61
	Vicuna-33B		0.54	0.25
	LLaMA-13B	1	0.57	0.43
	Vicuna-13B		0.42	0.29
	LLaMA-33B		0.57	0.42
	Vicuna-33B		0.42	0.25

Table 1: Baseline acceptance rate (α) at $k = 1$ on Alpaca and WMT datasets.

and sampling method, on acceptance rates. As shown in Table 1, on the unfine-tuned target model, i.e., LLaMA, the draft models generate candidates with good acceptance rates if the inference is performed on the Alpaca dataset with argmax sampling. However, fine-tuning the target model (Vicuna, a fine-tuned version of LLaMA), changing the dataset[4] or the sampling method can lead to a decrease in acceptance rates. In most cases, we did not observe a significant effect of changing the target model size on acceptance rates, while increasing the size of the draft model had a limited positive effect on it.

Fig. 3 illustrates the acceptance rates at different k using LLaMA-68M as a draft model, with LLaMA-160M results in Appendix D. As k increases, we observe a consistent improvement in acceptance rates across the different models and datasets. The α curves tend to converge when k exceeds 32, at which point it becomes difficult and uneconomical to increase k further. These results

[4]The discrepancy between the datasets can be attributed to the variation in prompt length: WMT averages 29 words per sample, compared to Alpaca's 9.

Dataset	Methods	Temp	LLaMA-33B k Config.	Speedup	τ	Vicuna-33B k Config.	Speedup	τ
Alpaca	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1x1x1	2.71×	3.19	1x1x1x1	1.64×	1.97
	SD (best γ)		1x1x1x1x1x1x1x1	2.89×	3.70	1x1x1x1x1x1	1.70×	2.03
	Ours		4x2x2x1x1	**3.06**×	**3.93**	8x2x1x1	**2.06**×	**2.56**
	SD (same γ)	1	1x1x1x1	1.69×	1.98	1x1x1	1.33×	1.53
	SD (best γ)		1x1x1x1x1	1.73×	2.02	1x1x1x1x1	1.34×	1.58
	Ours		8x2x1x1	**2.17**×	**2.71**	16x1x1	**1.73**×	**2.10**
WMT	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1x1x1	2.02×	2.38	1x1	1.15×	1.29
	SD (best γ)		1x1x1x1x1x1	2.06×	2.52	1x1	1.15×	1.29
	Ours		4x2x2x1x1	**2.24**×	**2.87**	16x2	**1.45**×	**1.73**
	SD (same γ)	1	1x1x1	1.41×	1.63	1x1	1.13×	1.26
	SD (best γ)		1x1x1x1x1	1.43×	1.69	1x1x1	1.14×	1.31
	Ours		8x2x1	**1.72**×	**2.08**	16x2	**1.42**×	**1.70**

Table 2: Performance of each method on Alpaca and WMT datasets using LLaMA-68M as draft model, and LLaMA-33B and Vicuna-33B as target models.

Dataset	Methods	Temp	LLaMA-33B k Config.	Speedup	τ	Vicuna-33B k Config.	Speedup	τ
Alpaca	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1x1	2.10×	3.20	1x1x1	1.33×	1.97
	SD (best γ)		1x1x1x1x1	2.16×	3.49	1x1	1.40×	1.81
	Ours		4x2x2x1	**2.42**×	**3.87**	8x2x2	**1.72**×	**2.59**
	SD (same γ)	1	1x1x1	1.43×	2.05	1x1	1.24×	1.60
	SD (best γ)		1x1	1.47×	1.90	1x1	1.24×	1.60
	Ours		8x2x2	**1.85**×	**2.79**	16x2	**1.62**×	**2.21**
WMT	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1x1	1.54×	2.42	1	1.06×	1.25
	SD (best γ)		1x1x1	1.57×	2.25	1	1.06×	1.25
	Ours		8x2x1x1	**1.81**×	**2.96**	32	**1.33**×	**1.66**
	SD (same γ)	1	1x1	1.22×	1.62	1	1.06×	1.25
	SD (best γ)		1x1	1.22×	1.62	1	1.06×	1.25
	Ours		16x2	**1.52**×	**2.14**	32	**1.30**×	**1.63**

Table 3: Performance of each method on Alpaca and WMT datasets using LLaMA-160M as draft model, and LLaMA-33B and Vicuna-33B as target models.

demonstrate the effectiveness of our method in improving the acceptance rate with only a small value of k.

4.3 Main Results

Given a draft of size γ, our method samples multiple tokens at each step, assuming they are $k_1, k_1, \cdots, k_\gamma$ respectively, generating a total of $K = \prod_{i=1}^{\gamma} k_i$ candidates. This constitutes a huge search space of hyperparameter. For efficiency reasons, we restrict the total budget K to a maximum of 32, and $k_i \in \{1, 2, 4, 8, 16, 32\}$. For the performance of our method under different hyperparameters, see Section 4.4. For each setting, we report the performance of our method for the best combination of ks, as well as the SD with the same γ and the SD with the best γ. We place the results of 13B versions in Appendix E to save space.

Table 2 shows the performance of each method using LLaMA-68m as the draft model. SD accelerates the LLaMA-33b model well on the Alpca dataset with argmax sampling, while the boost of our method is limited. However, when we observe a degradation in acceptance rate, as seen when using Vicuna as the target model, employing standard sampling, or working with the WMT dataset, the performance of SD degrades significantly. In these scenarios, our method demonstrates considerable improvement over SD. Furthermore, the difference observed between SD (same γ) and SD (best γ)

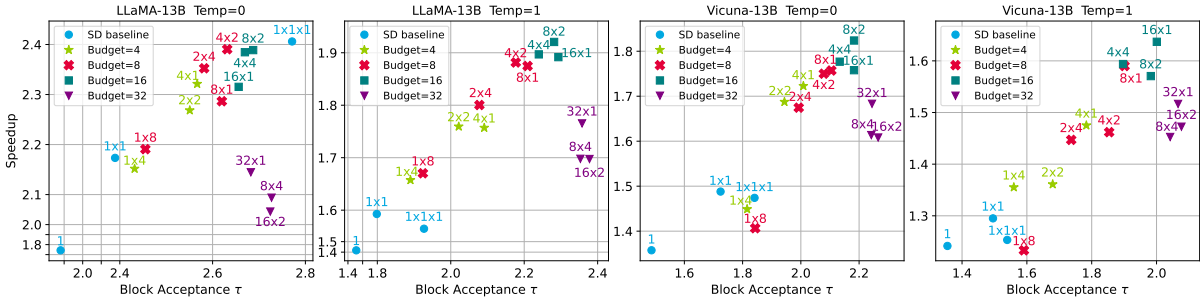


Figure 4: Speedup and block efficiency (τ) for different k configurations, where the dataset is Alpaca, using LLaMA-68M as the draft model.

Algorithm	Tree Attn.	LLaMA-13B Speedup	LLaMA-13B τ	Vicuna-13B Speedup	Vicuna-13B τ	LLaMA-33B Speedup	LLaMA-33B τ	Vicuna-33B Speedup	Vicuna-33B τ
MCSS w/o Replacement.	✓	1.91×	2.45	1.64×	2.00	2.17×	2.71	1.73×	2.10
MCSS w/ Replacement.	✓	1.89×	2.39	1.55×	1.95	2.04×	2.55	1.66×	2.01
MCNS	✓	1.21×	1.52	1.45×	1.78	1.33×	1.65	1.50×	1.80
MCSS w/o Replacement.	✗	1.61×	2.47	1.36×	1.99	1.84×	2.68	1.51×	2.09

Table 4: Ablation experiments on Tree Attention and different verification algorithms, where the dataset is Alpaca, the draft model is LLaMA-68M, and temp = 1 (standard sampling).

validates our claim that further increasing γ does not yield significant gains.

Replacing the draft model with LLaMA-160m, the results are shown in Table 3. The notable difference from LLaMA-68m is that the latency of LLaMA-160m is much higher. Consequently, the higher cost of the invocations leads to a general shrinkage of γ. Our method achieves a similar improvement as in Table 2, and in some cases, our method can works at a larger γ compared to SD (best γ), because it compensates for the additional overhead of the draft model by improving the acceptance rate.

Overall, our approach consistently achieves higher speedup and block efficiency compared to and SD baseline, demonstrating the effectiveness in improving the efficiency of the target model.

4.4 Budget Configuration

In this section, we examine the performance variations under different budget configurations. Fig. 4 shows the performance of the 13B-sized target model with different k configurations on the Alpaca dataset, where we fixed $\gamma = 2$ for clarity.

Our analysis yields three key insights. Firstly, the monotonically decreasing configuration always outperforms the monotonically increasing configuration for the same budget (monotonic rule), e.g., 4x2 outperforms 2x4. This is because the acceptance of the next token is governed by the acceptance of the preceding tokens. Therefore, it is a natural strategy to use a monotonically decreasing configuration to preferentially improve the acceptance of the preceding tokens. The monotonic rule is practically useful in reducing the overhead in hyperparameter search. Secondly, configurations with the same budget have roughly the same performance if we follow the monotonic rule. For instance, the 16-budget group is roughly centrally distributed, as is the 32-budget group. This underscores the robustness of our approach given a specific budget. Lastly, despite a higher block efficiency, the 32-budget group demonstrates a lower speedup than the 16-budget group. This counterintuitive outcome stems from the diminishing returns of block efficiency gains as budget increases, which fail to compensate for the latency inherent to larger budgets.

4.5 Ablation Study

We conduct an ablation study to investigate the impact of our proposed improvements on performance, given optimal k configurations as in Section 4.3. Our focus is on Tree Attention and verification algorithms that handle multiple candidates. More specifically, we look at multi-candidate speculative sampling (MCSS) both with and without replacement, as well as a multi-candidate variant of naive sampling (MCNS)[5]. The experiments are

[5]MCNS first sample a token x from $p(x)$ and accepts $\tilde{x}$ if $x \in \{\tilde{x}^1, \cdots, \tilde{x}^k\}$, otherwise it rejects the candidates and returns x.

based on standard sampling. For argmax sampling, MCSS with replacement degenerates to SD, as sampling with temp $= 0$ consistently produces the top1 token. Additionally, it can be conclusively demonstrated that MCSS without replacement is equivalent to MCNS when using argmax sampling.

The findings, as presented in Table 4, reveal that MCSS offers the most significant improvement when compared to MCNS. Tree Attention also contributes a crucial role, not altering the expected block efficiency but substantially reducing the communication overhead. The improvement brought by MCSS without replacement is limited, probably because repeated sampling is not as likely to happen when k_i is small.

5 Generalization Across Models

We evaluate the effectiveness of our method across a range of draft and target models. For draft models, we explore the intuitive hypothesis that fine-tuning the draft model would enhance the alignment with target models. We also examine the compatibility of our method when applied to the fine-tuned draft models. To accomplish this, we use ShareGPT data to fine-tune the LLaMA-68M and LLaMA-160M models, following the training setup provided by the Vicuna suite (Chiang et al., 2023). The resulting fine-tuned models are subsequently named Vicuna-68M and Vicuna-160M. With regard to the target models, our consideration extended to the LLaMA2 suite (Touvron et al., 2023b) and the OPT suite (Zhang et al., 2022; Iyer et al., 2022).

We report $\alpha@k$ with k from 1 to 4 for each model pair, as shown in Table 7. Fine-tuned draft models, Vicuna-68/160M, exhibit better alignment with both the Vicuna model and the LLaMA2-chat model, with a slight loss of alignment to the LLaMA/LLaMA2 base models. Fine-tuning, however, shows domain-specific limitations, yielding less improvement in baseline acceptance on the WMT dataset compared to the Alpaca dataset. The delta value suggests that our method is fully superimposable with fine-tuning, and even surpasses the improvements achieved prior to fine-tuning. In the context of the OPT suite, our approach achieves peak enhancements in acceptance rates for the OPT-iml-30B model on the WMT dataset.

6 Related Work

The quest for enhancing the inference efficiency of deep neural networks encompasses a broad spectrum of strategies, including but not limited to distillation (Hinton et al., 2015), quantisation (Dettmers et al., 2022), and sparcification (Jaszczur et al., 2021). These techniques often introducing some level of loss. In contrast, speculative decoding, as introduced by Leviathan et al., 2023 and Chen et al., 2023, effectively reduces the inference latency of LLMs without compromising model performance. Before them, blockwise parallel decoding (Stern et al., 2018) speeds up the inference of autoregressive models based on a similar principle, but it only supports argmax decoding. And Xia et al., 2023 shares similar idea, but uses a lossy verification algorithm.

Given the pivotal role of distributional consistency between the draft and target models, researchers have focused on aligning the draft model with the target model through additional training (Miao et al., 2023; Zhou et al., 2023; Liu et al., 2023). However, our empirical findings suggest that this alignment is less robust on out-of-distribution data (WMT) compared to in-distribution data (Alpaca). In line with our research, some studies have employed multiple candidates to improve the acceptance rate. Miao et al., 2023 utilises multiple draft models to generate diverse candidates, while Cai et al., 2023 trains additional prediction heads for the same purpose. Their work also incorporates tree attention to reduce the communication overhead associated with multiple candidates. Similar to our approach, Sun et al., 2023 also sample multiple candidates from the draft model. The difference is that they derive the algorithm for multi-candidate verification from the perspective of optimal transport, which requires linear programming for its implementation.

7 Conclusion

This paper introduces multi-candidate speculative decoding. This method leverages the full potential of multiple candidates generated by the draft model, thereby improving the acceptance rate without compromising the output quality of the target model. We further augment this approach with Tree Attention that reduces communication overhead. Extensive testing across various models, decoding settings, and datasets has shown that our method consistently reduces latency compared to standard speculative decoding. Our method works out-of-the-box and also benefits from works that improve acceptance rates with additional training.

Limitations

Our method, including speculative decoding, demands additional computational resources for parallel verification compared to incremental decoding. Consequently, the acceleration may be diminished when decoding multiple sequences simultaneously, which depends on the parallel computing capabilities of the device. On the other hand, the optimal hyperparameter configuration for our method is dependent on the model, dataset, and decoding settings. While our experiments indicate robust performance under the same budget, fut ure work can involve more efficiently determining the optimal hyperparameter configuration.

References

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

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907.

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

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

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

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

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

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

David R So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. 2021. Primer: Searching for efficient transformers for language modeling. *arXiv preprint arXiv:2109.08668*.

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

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023. Spectr: Fast speculative decoding via optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

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

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

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

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

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

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

A Proof of Multi-Candidate Speculative Sampling

Given any distributions $p(x)$ and $q(x)$, we now prove that the token returned from Algorithm 1 are distributed identically to those sampled from $p(x)$ alone.

Proof. Let $\tilde{x}^1, \cdots, \tilde{x}^k$ be the candidate tokens sampled from $q(x)$. We define

$$\begin{aligned} &\mathcal{R}_0 := \emptyset, \\ &\mathcal{R}_n := \text{Reject } \tilde{x}^1, \cdots, \tilde{x}^n, \end{aligned} \tag{2}$$

and

$$\begin{aligned} &\hat{p}_0(x) := p(x), \\ &\hat{p}_n(x) := \frac{\max(\hat{p}_{n-1}(x) - q(x), 0)}{\sum_x \max(\hat{p}_{n-1}(x) - q(x), 0)}. \end{aligned} \tag{3}$$

According to Algorithm 1, there is

$$P(\text{Accept } \tilde{x}^n | \tilde{x}^n = x, \mathcal{R}_{n-1}) = \min(1, \frac{\hat{p}_{n-1}(x)}{q(x)}). \tag{4}$$

Thus we have

$$\begin{aligned} &P(x, \text{Accept } \tilde{x}^n | \mathcal{R}_{n-1}) \\ &= P(\text{Accept } \tilde{x}^n, \tilde{x}^n = x | \mathcal{R}_{n-1}) \\ &= P(\tilde{x}^n = x | \mathcal{R}_{n-1}) P(\text{Accept } \tilde{x}^n | \tilde{x}^n = x, \mathcal{R}_{n-1}) \\ &= P(\tilde{x}^n = x) P(\text{Accept } \tilde{x}^n | \tilde{x}^n = x, \mathcal{R}_{n-1}) \\ &= q(x) \min(1, \frac{\hat{p}_{n-1}(x)}{q(x)}) \\ &= \min(q(x), \hat{p}_{n-1}(x)). \end{aligned} \tag{5}$$

The probability of rejecting $\tilde{x}^n$ is

$$\begin{aligned} &P(\text{Reject } \tilde{x}^n | \mathcal{R}_{n-1}) = 1 - P(\text{Accept } \tilde{x}^n | \mathcal{R}_{n-1}) \\ &= 1 - \sum_x P(\text{Accept } \tilde{x}^n, \tilde{x}^n = x | \mathcal{R}_{n-1}) \\ &= 1 - \sum_x \min(q(x), \hat{p}_{n-1}(x)) \\ &= \sum_x \max(\hat{p}_{n-1}(x) - q(x), 0). \end{aligned} \tag{6}$$

Let $A_n := P(x | \mathcal{R}_n)$. We have

$$\begin{aligned} &A_n = P(x | \mathcal{R}_n) \\ &= P(x, \text{Accept } \tilde{x}^{n+1} | \mathcal{R}_n) + P(x, \text{Reject } \tilde{x}^{n+1} | \mathcal{R}_n) \\ &= P(x, \text{Accept } \tilde{x}^{n+1} | \mathcal{R}_n) \\ &\quad + P(\text{Reject } \tilde{x}^{n+1} | \mathcal{R}_n) P(x | \mathcal{R}_{n+1}) \\ &= q(x) \min(1, \frac{\hat{p}_n(x)}{q(x)}) + \sum_x \max(\hat{p}_n(x) - q(x), 0) A_{n+1} \\ &= \min(q(x), \hat{p}_n(x)) + \sum_x \max(\hat{p}_n(x) - q(x), 0) A_{n+1}. \end{aligned} \tag{7}$$

According to Algorithm 1, $A_k = P(x|\mathcal{R}_k) = \hat{p}_k(x)$. So we get the recursive formula

$$\begin{aligned}
A_{k-1} &= \min(q(x), \hat{p}_{k-1}(x)) \\
&\quad + \sum_x \max(\hat{p}_{k-1}(x) - q(x), 0) A_k \\
&= \min(q(x), \hat{p}_{k-1}(x)) \\
&\quad + \sum_x \max(\hat{p}_{k-1}(x) - q(x), 0) \hat{p}_k(x) \\
&= \min(q(x), \hat{p}_{k-1}(x)) \\
&+ \sum_x \max(\hat{p}_{k-1}(x) - q(x), 0) \frac{\max(\hat{p}_{k-1}(x) - q(x), 0)}{\sum_x \max(\hat{p}_{k-1}(x) - q(x), 0)} \\
&= \min(q(x), \hat{p}_{k-1}(x)) + \max(\hat{p}_{k-1}(x) - q(x), 0) \\
&= \hat{p}_{k-1}(x).
\end{aligned} \tag{8}$$

Iteratively, we have

$$\begin{aligned}
A_0 = \hat{p}_0(x) = p(x), \\
P(x) = P(x|\mathcal{R}_0) = A_0 = p(x).
\end{aligned} \tag{9}$$

□

B Proof of Multi-Candidate Speculative Sampling without Replacement

Given any distributions $p(x)$ and $q(x)$, we now prove that the token returned from Algorithm 2 are distributed identically to those sampled from $p(x)$ alone.

Proof. Let $\tilde{x}^1, \cdots, \tilde{x}^k$ be the candidate tokens sampled without replacement from q, as in Eq. (1). We define

$$\begin{aligned}
\bar{p}_0(x) &= p(x), \\
\bar{p}_n(x) &= \frac{\max(\bar{p}_{n-1}(x) - \bar{q}_{n-1}(x), 0)}{\sum_x \max(\bar{p}_{n-1}(x) - \bar{q}_{n-1}(x), 0)}.
\end{aligned} \tag{10}$$

According to Algorithm 2, there is

$$\begin{aligned}
&P(\text{Accept } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^{n+1}) \\
&\qquad = \min(1, \frac{\bar{p}_n(\tilde{x}^{n+1})}{\bar{q}_n(\tilde{x}^{n+1})}).
\end{aligned} \tag{11}$$

Here we reuse the definition of $\mathcal{R}$ from Eq. (2). Thus we have

$$\begin{aligned}
&P(x, \text{Accept } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= P(\tilde{x}^{n+1} = x, \text{Accept } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= P(\tilde{x}^{n+1} = x|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&\quad \times P(\text{Accept}\tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^{n+1}) \\
&= \bar{q}_n(x) \min(1, \frac{\bar{p}_n(x)}{\bar{q}_n(x)}) \\
&= \min(\bar{q}_n(x), \bar{p}_n(x)).
\end{aligned} \tag{12}$$

and

$$\begin{aligned}
&P(x, \text{Reject } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= \sum_{\tilde{x}^{n+1}} P(x, \text{Reject } \tilde{x}^{n+1}, \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= \sum_{\tilde{x}^{n+1}} [P(\tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&\quad \times P(x, \text{Reject } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^{n+1})] \\
&= \sum_{\tilde{x}^{n+1}} [P(\tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&\quad \times P(\text{Reject } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^{n+1}) \\
&\quad \times P(x|\mathcal{R}_{n+1}, \tilde{x}^1, \cdots, \tilde{x}^{n+1})] \\
&= \sum_{\tilde{x}^{n+1}} [P(\tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&\quad \times (1 - P(\text{Accept } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^{n+1})) \\
&\quad \times P(x|\mathcal{R}_{n+1}, \tilde{x}^1, \cdots, \tilde{x}^{n+1})] \\
&= \sum_{\tilde{x}^{n+1}} [\bar{q}_n(\tilde{x}^{n+1})(1 - \min(1, \frac{\bar{p}_n(\tilde{x}^{n+1})}{\bar{q}_n(\tilde{x}^{n+1})})) \\
&\quad \times P(x|\mathcal{R}_{n+1}, \tilde{x}^1, \cdots, \tilde{x}^{n+1})] \\
&= \sum_{\tilde{x}^{n+1}} \{[\bar{q}_n(\tilde{x}^{n+1}) - \min(\bar{q}_n(\tilde{x}^{n+1}), \bar{p}_n(\tilde{x}^{n+1}))] \\
&\quad \times P(x|\mathcal{R}_{n+1}, \tilde{x}^1, \cdots, \tilde{x}^{n+1})\}.
\end{aligned} \tag{13}$$

Let $A_n := P(x|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n)$. We have

$$\begin{aligned}
A_n &= P(x|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= P(x, \text{Accept } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&\quad + P(x, \text{Reject } \tilde{x}^{n+1}|\mathcal{R}_n, \tilde{x}^1, \cdots, \tilde{x}^n) \\
&= \min(\bar{q}_n(x), \bar{p}_n(x)) \\
&+ \sum_{\tilde{x}^{n+1}} \{[\bar{q}_n(\tilde{x}^{n+1}) - \min(\bar{q}_n(\tilde{x}^{n+1}), \bar{p}_n(\tilde{x}^{n+1}))] A_{n+1}\}.
\end{aligned} \tag{14}$$

Because $\bar{q}_n(x)$ is independent of $\tilde{x}^{n+1}$. Thus $\bar{p}_n$ is independent of x_n holds for $n = 1, \cdots, k$. According to Algorithm 2,

$$A_k = P(x|\mathcal{R}_k, \tilde{x}^1, \cdots, \tilde{x}^k) = \bar{p}_k(x), \tag{15}$$

so A_k is also independent of $\tilde{x}^k$. We have

$$
\begin{aligned}
&A_{k-1} \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ \sum_{\tilde{x}^k}\{[\bar{q}_{k-1}(\tilde{x}^k) - \min(\bar{q}_{k-1}(\tilde{x}^k), \bar{p}_{k-1}(\tilde{x}^k))]A_k\} \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ A_k \sum_{\tilde{x}^k}[\bar{q}_{k-1}(\tilde{x}^k) - \min(\bar{q}_{k-1}(\tilde{x}^k), \bar{p}_{k-1}(\tilde{x}^k))] \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ A_k[\sum_{\tilde{x}^k} \bar{q}_{k-1}(\tilde{x}^k) - \sum_{\tilde{x}^k} \min(\bar{q}_{k-1}(\tilde{x}^k), \bar{p}_{k-1}(\tilde{x}^k))] \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ A_k[\sum_{\tilde{x}^k} \bar{p}_{k-1}(\tilde{x}^k) - \sum_{\tilde{x}^k} \min(\bar{q}_{k-1}(\tilde{x}^k), \bar{p}_{k-1}(\tilde{x}^k))] \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ A_k \sum_{\tilde{x}^k}[\bar{p}_{k-1}(\tilde{x}^k) - \min(\bar{q}_{k-1}(\tilde{x}^k), \bar{p}_{k-1}(\tilde{x}^k))] \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ A_k \sum_{\tilde{x}^k} \max(\bar{p}_{k-1}(\tilde{x}^k) - \bar{q}_{k-1}(\tilde{x}^k), 0) \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) \\
&+ [\frac{\max(\bar{p}_{k-1}(x) - \bar{q}_{k-1}(x), 0)}{\sum_x \max(\bar{p}_{k-1}(x) - \bar{q}_{k-1}(x), 0)} \\
&\quad \times \sum_{\tilde{x}^k} \max(\bar{p}_{k-1}(\tilde{x}^k) - \bar{q}_{k-1}(\tilde{x}^k), 0)] \\
&= \min(\bar{q}_{k-1}(x), \bar{p}_{k-1}(x)) + \max(\bar{p}_{k-1}(x) - \bar{q}_{k-1}(x), 0) \\
&= \bar{p}_{k-1}(x).
\end{aligned}
\tag{16}
$$

Iteratively, we have

$$
\begin{aligned}
A_0 &= \bar{p}_0(x) = p(x), \\
P(x) &= P(x|\mathcal{R}_0) = A_0 = p(x).
\end{aligned}
\tag{17}
$$

□

C Upper Bound on Acceptance Rate for Naive Sampling

We show that for any $k \in \mathcal{N}^+$, the acceptance rate of multi-candidate speculative sampling (with replacement) has an upper bound for multi-candidate naive sampling. Our proof references Miao et al., 2023.

Proof. We use P_S and P_N to denote the probabilities involved in speculative sampling and naive sampling algorithms, respectively.

Since the acceptance rate is equal to $1 - \sum_x P(x, \mathcal{R}_k)$, where we reuse the definition of $\mathcal{R}$ from Eq. (2). We now prove that $P_S(x, \mathcal{R}_k) \leq P_N(x, \mathcal{R}_k)$ always holds.

For speculative sampling, we have

$$
\begin{aligned}
&P_S(x, \mathcal{R}_k) \\
&= P(x|\mathcal{R}_k)P(\mathcal{R}_k) \\
&= \hat{p}_k(x) \prod_{i=1}^{k} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}).
\end{aligned}
\tag{18}
$$

For naive sampling, there is

$$
P_N(x, \mathcal{R}_k) = p(x)(1 - q(x))^k. \tag{19}
$$

If there is $\hat{p}_i(x) - q(x) \leq 0$, then $\hat{p}_j(x) = 0$ holds for $j \geq i$. Thus $P_S(x, \mathcal{R}_k) = 0 \leq P_N(x, \mathcal{R}_k)$. Otherwise $\hat{p}_i(x) - q(x) > 0$ holds for $i = 0, \cdots, k-1$, and we get

$$
\hat{p}_i(x) = \frac{\hat{p}_{i-1}(x) - q(x)}{\sum_x \max(\hat{p}_{i-1}(x) - q(x), 0)} \tag{20}
$$

By mathematical induction, if $k = 1$, there is

$$
\begin{aligned}
&P_S(x, \mathcal{R}_1) = \hat{p}_1(x)P(\text{Reject } \tilde{x}^1|\mathcal{R}_0) \\
&= \frac{\hat{p}_0(x) - q(x)}{\sum_x \max(\hat{p}_0(x) - q(x), 0)} \sum_x \max(\hat{p}_0(x) - q(x), 0) \\
&= p(x) - q(x) \\
&\leq p(x) - p(x)q(x) \\
&= p(x)(1 - q(x)) \\
&= P_N(x, \mathcal{R}_1),
\end{aligned}
\tag{21}
$$

where the value of $P(\text{Reject } \tilde{x}^1|\mathcal{R}_0)$ comes from Eq. (6).

Assume that $P_S(x, \mathcal{R}_k) \leq P_N(x, \mathcal{R}_k)$ holds for $k < n$, then

$$
\begin{aligned}
&P_S(x, \mathcal{R}_n) = \hat{p}_n(x) \prod_{i=1}^{n} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&= \hat{p}_n(x)P(\text{Reject } \tilde{x}^n|\mathcal{R}_{n-1}) \prod_{i=1}^{n-1} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&= \frac{\hat{p}_{n-1}(x) - q(x)}{\sum_x \max(\hat{p}_{n-1}(x) - q(x), 0)} \sum_x \max(\hat{p}_{n-1}(x) - q(x), 0) \\
&\quad \times \prod_{i=1}^{n-1} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&= [\hat{p}_{n-1}(x) - q(x)] \prod_{i=1}^{n-1} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&\leq [\hat{p}_{n-1}(x) - \hat{p}_{n-1}(x)q(x)] \prod_{i=1}^{n-1} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&= [1 - q(x)]\hat{p}_{n-1}(x) \prod_{i=1}^{n-1} P(\text{Reject } \tilde{x}^i|\mathcal{R}_{i-1}) \\
&\leq [1 - q(x)]p(x)(1 - q(x))^{n-1} \\
&\leq p(x)(1 - q(x))^n \\
&= P_N(x, \mathcal{R}_n),
\end{aligned}
\tag{22}
$$

□

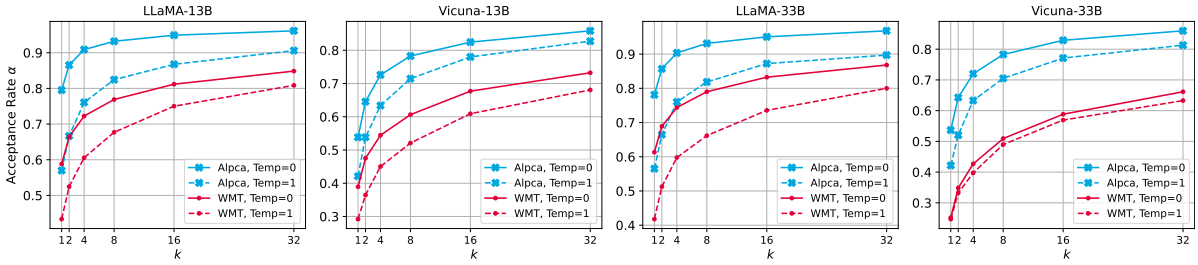


Figure 5: Acceptance rate (α) curves given different k using LLaMA-160M as draft model.

Dataset	Methods	Temp	LLaMA-13B k Config.	LLaMA-13B Speedup	LLaMA-13B τ	Vicuna-13B k Config.	Vicuna-13B Speedup	Vicuna-13B τ
Alpaca	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1x1x1	2.46×	3.35	1x1	1.49×	1.72
	SD (best γ)	0	1x1x1x1x1x1x1	2.57×	3.72	1x1	1.49×	1.72
	Ours	0	2x2x2x1x1	**2.75**×	**3.89**	8x2	**1.82**×	**2.18**
	SD (same γ)	1	1x1x1	1.54×	1.93	1x1	1.30×	1.50
	SD (best γ)	1	1x1	1.56×	1.79	1x1	1.30×	1.50
	Ours	1	4x2x2	**1.91**×	**2.45**	16x1	**1.64**×	**2.00**
WMT	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1	1.69×	2.05	1x1	1.29×	1.52
	SD (best γ)	0	1x1x1	1.69×	2.05	1x1x1x1	1.31×	1.63
	Ours	0	4x2x1	**1.88**×	**2.37**	8x1	**1.54**×	**1.85**
	SD (same γ)	1	1x1	1.30×	1.55	1x1	1.14×	1.34
	SD (best γ)	1	1x1x1	1.31×	1.63	1	1.15×	1.25
	Ours	1	8x2	**1.57**×	**1.95**	16x1	**1.39**×	**1.72**

Table 5: Performance of each method on Alpaca and WMT datasets using LLaMA-68M as draft model, and LLaMA-13B and Vicuna-13B as target models.

Dataset	Methods	Temp	LLaMA-13B k Config.	LLaMA-13B Speedup	LLaMA-13B τ	Vicuna-13B k Config.	Vicuna-13B Speedup	Vicuna-13B τ
Alpaca	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1x1	1.53×	2.91	1x1	1.15×	1.83
	SD (best γ)	0	1x1x1	1.53×	2.91	1	1.16×	1.54
	Ours	0	4x2x2	**1.70**×	**3.36**	4x4	**1.39**×	**2.25**
	SD (same γ)	1	1x1	1.17×	1.90	1	1.09×	1.42
	SD (best γ)	1	1	1.19×	1.57	1	1.09×	1.42
	Ours	1	8x2	**1.41**×	**2.37**	32	**1.29**×	**1.83**
WMT	Baseline	N/A	N/A	1×	1	N/A	1×	1
	SD (same γ)	0	1x1	1.18×	1.94	1	1.07×	1.39
	SD (best γ)	0	1x1	1.18×	1.94	1	1.07×	1.39
	Ours	0	16x1	**1.34**×	**2.31**	16	**1.25**×	**1.68**
	SD (same γ)	1	1	1.04×	1.43	1	0.99×	1.29
	SD (best γ)	1	1	1.04×	1.43	1	0.99×	1.29
	Ours	1	32	**1.27**×	**1.81**	32	**1.20**×	**1.68**

Table 6: Performance of each method on Alpaca and WMT datasets using LLaMA-160M as draft model, and LLaMA-13B and Vicuna-13B as target models.

D Acceptance Rate Improvement

Fig. 5 shows the acceptance rate improvement from increasing k when using LLaMA-160M as a draft model.

E Main Results for 13B Models

Table 5 and Table 6 shows the performance of each method when using target models of size 13B.

Draft model	Target model	Temp	$\alpha@1 \rightarrow \alpha@4$ ($\Delta\alpha$) Alpaca	WMT
LLaMA-68M	LLaMA-13B	0	0.76 → 0.88 (+0.12)	0.55 → 0.68 (+0.13)
	Vicuna-13B	0	0.49 → 0.67 (+0.19)	0.36 → 0.51 (+0.14)
	LLaMA2-13B	0	0.75 → 0.88 (+0.13)	0.57 → 0.71 (+0.14)
	LLaMA2-13B-chat	0	0.47 → 0.66 (+0.19)	0.30 → 0.44 (+0.15)
Vicuna-68M	LLaMA-13B	0	0.75 → 0.90 (**+0.14**)	0.56 → 0.69 (+0.13)
	Vicuna-13B	0	0.56 → 0.76 (**+0.20**)	0.38 → 0.57 (**+0.19**)
	LLaMA2-13B	0	0.74 → 0.89 (**+0.14**)	0.56 → 0.69 (+0.13)
	LLaMA2-13B-chat	0	0.55 → 0.75 (**+0.21**)	0.32 → 0.53 (**+0.21**)
LLaMA-160M	LLaMA-13B	0	0.80 → 0.91 (+0.11)	0.59 → 0.72 (+0.13)
	Vicuna-13B	0	0.54 → 0.73 (+0.19)	0.39 → 0.54 (+0.15)
	LLaMA2-13B	0	0.78 → 0.90 (+0.12)	0.61 → 0.74 (+0.14)
	LLaMA2-13B-chat	0	0.52 → 0.72 (+0.19)	0.32 → 0.48 (+0.16)
Vicuna-160M	LLaMA-13B	0	0.78 → 0.91 (**+0.12**)	0.59 → 0.72 (+0.13)
	Vicuna-13B	0	0.62 → 0.81 (+0.19)	0.40 → 0.58 (**+0.18**)
	LLaMA2-13B	0	0.77 → 0.90 (**+0.13**)	0.59 → 0.72 (+0.13)
	LLaMA2-13B-chat	0	0.61 → 0.81 (**+0.20**)	0.33 → 0.54 (**+0.22**)
OPT-125M	OPT-13B	0	0.86 → 0.95 (+0.09)	0.97 → 0.99 (+0.02)
	OPT-30B	0	0.83 → 0.94 (+0.11)	0.80 → 0.89 (+0.09)
	OPT-iml-30B	0	0.81 → 0.93 (+0.12)	0.40 → 0.77 (+0.37)
LLaMA-68M	LLaMA-13B	1	0.51 → 0.72 (+0.21)	0.39 → 0.56 (+0.17)
	Vicuna-13B	1	0.35 → 0.57 (+0.22)	0.25 → 0.41 (+0.16)
	LLaMA2-13B	1	0.51 → 0.71 (+0.20)	0.38 → 0.57 (+0.18)
	LLaMA2-13B-chat	1	0.35 → 0.57 (+0.22)	0.25 → 0.39 (+0.14)
Vicuna-68M	LLaMA-13B	1	0.48 → 0.69 (+0.21)	0.38 → 0.55 (**+0.18**)
	Vicuna-13B	1	0.46 → 0.68 (+0.22)	0.29 → 0.45 (+0.16)
	LLaMA2-13B	1	0.49 → 0.69 (+0.20)	0.38 → 0.56 (+0.18)
	LLaMA2-13B-chat	1	0.46 → 0.69 (+0.22)	0.30 → 0.49 (**+0.18**)
LLaMA-160M	LLaMA-13B	1	0.57 → 0.76 (+0.19)	0.43 → 0.61 (+0.17)
	Vicuna-13B	1	0.42 → 0.63 (+0.21)	0.29 → 0.45 (+0.16)
	LLaMA2-13B	1	0.57 → 0.75 (+0.19)	0.43 → 0.61 (+0.18)
	LLaMA2-13B-chat	1	0.41 → 0.63 (+0.22)	0.29 → 0.44 (+0.15)
Vicuna-160M	LLaMA-13B	1	0.54 → 0.73 (**+0.20**)	0.41 → 0.58 (**+0.18**)
	Vicuna-13B	1	0.53 → 0.74 (**+0.22**)	0.31 → 0.48 (**+0.18**)
	LLaMA2-13B	1	0.54 → 0.73 (+0.19)	0.41 → 0.59 (+0.18)
	LLaMA2-13B-chat	1	0.54 → 0.75 (+0.22)	0.33 → 0.51 (**+0.18**)
OPT-125M	OPT-13B	1	0.63 → 0.81 (+0.18)	0.59 → 0.78 (+0.19)
	OPT-30B	1	0.60 → 0.79 (+0.18)	0.56 → 0.76 (+0.20)
	OPT-iml-30B	1	0.61 → 0.78 (+0.17)	0.44 → 0.68 (+0.24)

Table 7: Acceptance rate (α) improvements with k from 1 to 4 on Alpaca and WMT datasets using various draft and target models. Underlining highlights $\alpha@k$ increases after fine-tuning, while **bold** indicates $\Delta\alpha$ improvements.