Improving Multi-candidate Speculative Decoding

Xiaofan Lu[*, 1], **Yixiao Zeng**[*, 2], **Feiyang Ma**[3], **Zixu Yu**[4], **Marco Levorato**[5]
University of California, Irvine
{[1]xiaofl14, [2]yixiaz8, [3]feiyangm, [4]zixuy, [5]levorato}@uci.edu

Abstract

Speculative Decoding (SD) is a technique to accelerate the inference of Large Language Models (LLMs) by using a lower complexity draft model to propose candidate tokens verified by a larger target model. To further improve efficiency, Multi-Candidate Speculative Decoding (MCSD) improves upon this by sampling multiple candidate tokens from the draft model at each step and verifying them in parallel, thus increasing the chances of accepting a token and reducing generation time. Existing MCSD methods rely on the draft model to initialize the multi-candidate sequences and use static length and tree attention structure for draft generation. However, such an approach suffers from the draft and target model's output distribution differences, especially in a dynamic generation context. In this work, we introduce a new version of MCSD that includes a target model initialized multi-candidate generation, a dynamic sliced topology-aware causal mask for dynamic length adjustment, and decision models to optimize early stopping. We experimented with our method on Llama 2-7B and its variants and observed a maximum **27.5%** speedup compared to our MCSD baseline across three benchmarks with Llama 2-7B as the target model and JackFram 68M as the draft model. Additionally, we evaluate the effects of using the target model initialized multi-candidate process with different draft models on output quality. Our original code is available on GitHub.

1 Introduction

In recent years, Large Language Models (LLMs) such as GPT-4[1] have significantly advanced various language processing tasks. However, these models are computationally intensive especially during the inference phase, where generating k tokens requires k serial runs of the model. This inefficiency limits the practical deployment of these powerful models in real-time applications.

Among serious LLM inference optimization methods, Speculative Decoding (SD)[4] has been shown to increase inference speed with a marginal generation quality loss. SD frameworks first generate candidate tokens using a model (the draft model) with lower complexity compared to the original model (the target model). Then, the target model verifies the generated tokens. The performance of SD is mainly determined by the token acceptance rate α, which measures the proportion of candidate tokens generated by the draft model that the target model accepts. The benefits of using a faster draft model diminish if the target model frequently rejects these tokens, which means that the target model must re-generate the tokens.

To further enhance the acceptance rate, Multi-Candidate Speculative Decoding (MCSD)[6, 10] was introduced. MCSD samples multiple candidate tokens at each generation step and verifies them in parallel using the target model. This approach increases the likelihood that at least one of the candidate tokens will be accepted, thereby improving the overall acceptance rate and efficiency. MCSD also incorporates a tree attention mechanism to manage computational and communication

[*]Contributed equally

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

overhead by organizing multiple candidate sequences into a single sequence and applying a carefully designed attention mask. However, MCSD still faces several challenges. **1. Increasing Computational Complexity**: Verifying multiple candidates simultaneously increases the computational load, requiring more memory and processing power. **2. Efficient Topology-Aware Causal Mask Generation**: Generating and maintaining a topology-aware causal mask for multi-candidate token trees is time-consuming and reduces the adaptivity of the model. **3. Fixed Draft Generation Length (γ)**: Using a fixed length for draft-generated token segments may not be optimal in all contexts.

In this paper, we present a method that introduces the dynamic sliced topology-aware causal mask to facilitate the speculative decoding process, allowing dynamic adjustment of the draft generation length without reconstructing the topology-aware causal mask. We enhance the acceptance rate by initializing the multi-candidate token tree with the target model, thus improving efficiency. Additionally, we incorporate a decision model to optimize the early stopping mechanism during the draft model generation stage. The model dynamically halts draft token generation early by predicting the likelihood of the target model accepting the tokens, thus reducing unnecessary computation.

Our experimental results show that our framework, the combination of target-initialized multi-candidate generation, dynamic sliced topology-aware causal mask, and early stop with a decision model, struggles to maintain both generation quality and speedup simultaneously. Instead, we found that our static target model initialized multi-candidate generation alone achieves the highest speedup while preserving the highest generation quality among our experiments. Therefore, we are presenting the speedup results from our static target-initialized multi-candidate generation experiments in section 4.1, and the results of our framework are in the Appendix A.1.

The static target model initialized multi-candidate generation method with the optimal multi-candidate generation configuration we find through grid search on custom dataset improves generation speed through the improvement in the acceptance rate (α), defined as the ratio of the longest draft sequence length accepted by the target model to the maximum draft sequence length. The method achieves a maximum of **27.5%** in generation speedup comparing with MCSD baseline and using smaller draft model (Llama-68M [6]) on three datasets: TriviaQA [3], Alpaca [8] and MT-Bench [12]. Output quality evaluation on MT-Bench reveals that the target model initialized multi-candidate process does not preserve the target model's output quality; the output quality decreases as the number of target-initialized tokens increases, and different draft models significantly impact output quality. We also conduct an ablation study to evaluate the impact of the decision model in Appendix A.2. Additionally, we analyze why our framework does not outperform static target-initialized multi-candidate generation alone in Session 5.

2 Background

2.1 Speculative Decoding

Speculative decoding is the collaboration of two models: a smaller draft model (often a more efficient approximation model) and a larger target model. First, the draft model generates multiple candidate tokens in parallel, using its probability distribution to predict the following possible tokens based on input. Then, these tokens are passed to the target model, which verifies them by computing their probability distribution over the same input. If the target model accepts the candidate tokens (probabilities alignment), they are finalized in the output sequence. Otherwise, the target model replaces them by generating new tokens based on its own distribution. The speculative decoding process ensures that the output distribution remains consistent with what the target model alone would produce, thus maintaining the quality of the generated content[4].

Importantly, this technique does not require changes to the model's architecture or retraining, making it an accessible and efficient solution for accelerating inference.

2.2 Multi-Candidate Generation

Due to distributional differences between the draft and target models, the candidate path with the highest probability in the draft model may not always result in the most accepted tokens by the target model. Therefore, verifying multiple candidate paths in parallel increases the overall acceptance rate α of draft tokens. This paper takes the tree attention methodology in Specinfer [6] as a starting point to process multiple candidate token paths concurrently. Unlike the traditional causal attention

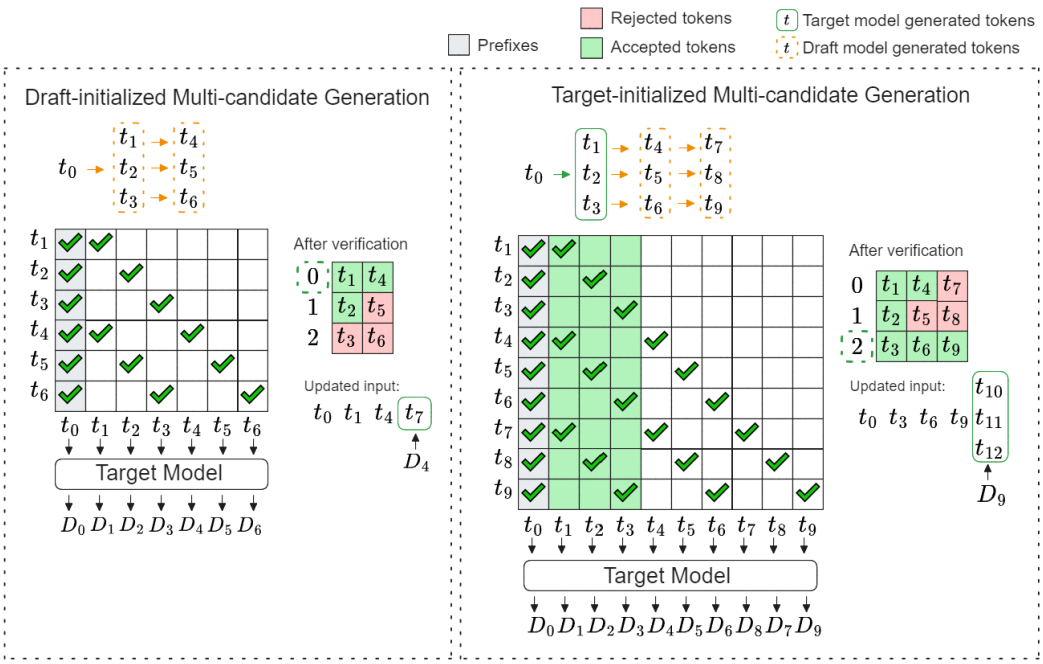

Figure 1: Both the draft-initialized (left) and target-initialized (right) multi-candidate generation processes utilize a token tree configuration with a width of 3 and depth of 2. The execution sequence proceeds as follows: (**1**) Generate the token tree (shown at the top of each diagram). (**2**) Transform the token tree into a topology-aware causal mask (represented as a square mask with a check symbol). (**3**) Generate multi-candidate sequences using the draft model (not shown in the figure). (**4**) Verify the multi-candidate sequences with the target model by obtaining next-token logits, which are then transformed into distributions. (**5**) Select the candidate sequence with the longest length after verification. (**6**) Update the input IDs, key-value cache, and sample new token(s) based on the target model's next-token distributions. Note: In a draft-initialized multi-candidate generation, only one new token is sampled, whereas in a target-initialized multi-candidate generation, multiple new tokens are sampled.

paradigm, which was designed for single text sequence generation, tree attention calculates attention scores for multiple text sequences from a token tree, which requires a topology-aware casual mask plus changes in usual single candidate positional indices and key-value cache update to fuse tree attention computation of all tokens in a single kernel. In Figure 1, we illustrate the topology-aware causal mask for calculating the tree attention of a multi-candidate sequence with three candidate token paths, each with two draft tokens.

3 Method

In the speculative decoding framework proposed by Leviathan et al. [4], the expected improvement factor (IF) is defined as:

$$IF = \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(c\gamma+1)}$$

where α represents the expectation of acceptance rate, γ represent the draft generation length, and c represent the ratio between the time for a single run of draft model and time for a single run of target model. Based on the formula (1), a larger α will speedup token generation. If an oracle could determine γ dynamically, the improvement factor can be up to around 60% larger than the improvement factor with a fixed γ.

In this work, we explore three methods to improve α and determine γ dynamically to accelerate the MCSD process:

1. Target Model Initialized Multi-Candidate Generation: we introduce a new methodology to construct multi-candidate sequences that improve α over existing approaches;
2. Dynamic Sliced Topology-Aware Causal Mask: we introduce a method to efficiently create topology-aware casual masks for dynamic multi-candidate generation;
3. Early Stop Decision Model: we introduce a low-complexity MLP model to determine γ dynamically during each draft generation loop.

We remark that the above processes can be integrated into a unified framework. The Target Model Initialized Multi-Candidate Token Tree method can be deployed in isolation as it improves α for both static and dynamic MCSD, while 2 and 3 are dependent on each other and need the dynamic MCSD.

3.1 Target Model Initialized Multi-Candidate Generation

Existing multi-candidate speculative decoding methods rely on a draft model to generate the entire multi-candidate token tree, and after verifying the draft generated tokens, only sample **one token** from the target model or normalized target and draft model's output distribution. Due to the difference between the output distribution of the draft and target model, there is no specific criterion to determine which token sampled from the target model will yield the longest accepted token sequence for future draft model generation. Therefore, we hypothesize that sampling multiple tokens instead of one token from the target model's distribution to initialize a multi-candidate sequence for future draft model generation can increase the acceptance rate. However, When the target model initializes more than one token for multi-candidate generation, selecting an initialized token based on the longest accepted draft token sequence creates a dependency. This means that the acceptance probabilities of the sequential draft tokens influence the probability of accepting the target-sampled token. Consequently, the output distribution no longer aligns with the target model's. Specifically, by selecting the target token t that yields the longest accepted draft sequence, the probability of accepting the target sampled token t becomes:

$$P_{\text{output}}(t) = P_{\text{target}}(t) \times P_{\text{acceptance}}(t+1, t+2, \cdots \mid t)$$

Where $P_{\text{acceptance}}(t+1, t+2, \cdots \mid t)$ is the probability that the subsequent draft tokens are accepted given the target token t. This alters the output distribution, causing it to diverge from $P_{\text{target}}(t)$. Thus, we evaluate the quality of our method's output using MT-Bench. The empirical results indicate that the quality loss depends on the number of target model initialized tokens and the draft model. For more details, please refer to the experimental section.

Figure 1 illustrates how we modified the existing draft-initialized multi-candidate generation to build our target model-initialized multi-candidate generation. The topology-aware mask for the target-initialized token tree needs a larger mask than the draft-initialized token tree to handle the multiple initial tokens for different possible sequences sampled from the target model. The topology-aware causal mask size for the target model initialized token tree is the square of the sum of the token tree width and the total number of draft-generated tokens. In Figure 1, the mask size for the target model initialized token tree is $(3+6)^2$ while the size for the draft initialized token tree is 6^2. We note that if draft-initialized and target-initialized multi-candidate generation has the same acceptance rate, then the target-initialized method will perform one more target forward pass compared to the draft-initialized method as the origin input passes into the target model instead of the draft model. However, in general, text generation tasks usually involve hundreds of target forward passes, and this overhead is marginal.

3.2 Dynamic Sliced Topology-Aware Causal Mask

Existing multi-candidate speculative decoding methods such as EAGLE [5] employ expansion-based token trees with various depths and widths for different branches to increase the target model's average acceptance rate on draft-generated tokens. Generating a topology-aware casual mask for the multi-candidate token tree is time-consuming; most existing multi-candidate speculative decoding methods only build the topology-aware casual mask once during initialization.

We introduce a dynamic sliced topology-aware casual mask to allow the decision model to dynamically decide the length of multi-candidate draft token generation and avoid generating a new topology-aware casual mask during each iteration. The main idea is to construct a large topology-aware casual mask during initialization. Figure 2 illustrates a topology-aware casual mask that allows a maximum

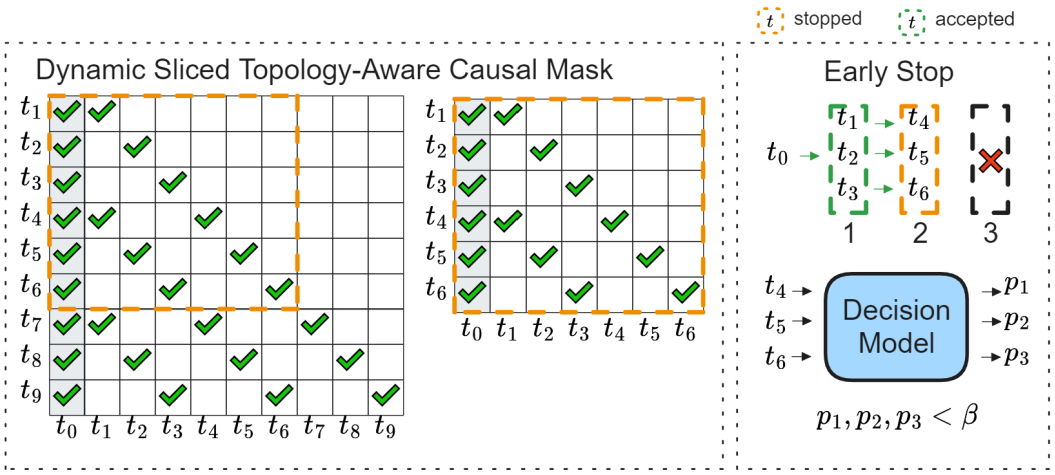

Figure 2: β denotes the threshold for early stop, in our experiment the $\beta = 0.4$. The inputs for the decision model are hidden states or output distribution and entropy related to the token. The dynamic multi-candidate speculative decoding process with an early stop decision model and fork-shaped draft model initialized a token tree, where the token tree configuration is $W = 3$ (width) and $D = 3$ (depth). It stopped at the second draft generation turn, where the maximum number of draft generation turns is three.

of three draft token generations for each candidate sequence. There are three possible sliced causal masks for dynamic multi-candidate speculative decoding: mask when early stop at first iteration (upper left 3 x 3 causal mask), early stop at second iteration (shown in Figure 2), no early stop (entire 9 x 9 casual mask).

In the proposed framework, we keep the token tree in fork shape so that each candidate token sequence does not expand to new sub-sequences. The main benefit of using a fork shape instead of an expansion-based token tree is a reduction in memory usage since the size of a topology-aware causal mask for an expansion-based token tree grows exponentially. In contrast, the fork-shaped token tree will grow linearly.

3.3 Decision Model

We design two types of decision models to dynamically determine if the early stop is necessary for the draft generation process. The first type of decision model is a three-layer MLP, taking the hidden states from draft model as input and the $max(1, \frac{p(x|I)}{q(x|I)})$ ($p(x|I)$ is the probability of target model predict token x with given input I and $q(x|I)$ is the probability of draft model predict token x with given input I) as the training label. The decision process can be represented as:

$$P_{T1} = MLP(y_i^H)$$

Where y_i^H denotes the hidden states of the draft model for token i. The second type of decision model is inspired by Tandem transformer [7], is a two layer-MLP with draft model's output distribution's entropy that takes probabilities as input and the result of verification (zero as rejected, one as accepted) as a label. The decision process can be represented as:

$$P_{T2} = MLP(ConCat(Topk(y_i^D), Ent(y_i^D))$$

Where y_i^D represents the output distribution of the draft model for token i and is used to compute both the entropy and top-k probabilities for input to the two layer-MLP.

During the draft generation process, the decision model will batch inference the input for multiple sequences and calculate the probabilities of the target model accepting each sequence. If the probabilities for all sequences are lower than the threshold (in our experiment, the threshold is 0.4 for both decision models), then draft generation will stop early; otherwise, it will continue until the maximum draft generation length is reached.

4 Experiment

4.1 Experimental Setup

Our experiments are performed on a single server equipped with an Nvidia RTX 4090 GPU. To ensure consistency with the original training parameters of the Llama model, all target models use Bfloat16 precision, while draft models are configured with double precision. This setup provides a stable testing environment: Bfloat16 precision on draft models occasionally leads to NaN or Inf values in softmax calculations within the PyTorch library, but double precision for draft models prevents these issues from interrupting the experiment.

For our testing dataset, we select TriviaQA [3], Alpaca [8] and MT-Bench [12]. In the following testing, for TriviaQA and Alpaca, we randomly select 250 input prompts. And for MT-bench, we test the complete 80 sets of prompts.

For the draft and target models, we extensively utilize the Llama 2-7B Chat model [9] and other related models that share the same tokenizer. As for our target models, we select LLama 2-7B Chat and its fine-tuned version, Vicuna-7B [2]. At the same time, we use two different models, the Llama-68M from SpecInfer [6] and the TinyLlama-1.1B [11], as draft models to test the effect of the acceleration methods on different types of model pairs. The maximum generation length is 200 tokens. We apply a temperature of 0 for greedy sampling and 0.7 for probabilistic sampling, with the latter value providing a midpoint that balances the trade-off between generation speed and quality, as observed in our experiments.

Furthermore, we standardize the configuration across different SD methods to evaluate the acceptance rate and other performance metrics. Specifically, we set the SD to a fixed $\gamma = 4$, while for the MCSD method, we use the optimal k-configuration of (4,2,2,1) in our environment, as it provides a greater speedup than the (4,2,2,1,1) configuration reported in the original paper [10].

Our method employs a static target model initialized MCSD configuration with (2,4,3,1,1), where the first number (2) in the configuration represents the number of target model initialized tokens, and the draft model will generate $4 + 4 \times 3 + 4 \times 3 \times 1 + 4 \times 3 \times 1 \times 1 = 40$ draft tokens for each target model initialized token result $2 \times 4 \times 3 \times 1 \times 1 = 24$ different candidate sequences with draft token length equal to four ($\gamma = 4$). Figure 3 shows the speedup ratio under such circumstances. Table 1 shows that our method achieves the highest output quality when the number of target-initialized tokens is set to two. Notably, when the number of target-initialized tokens is reduced to one, the target output distribution remains stable, and the MT-bench score is preserved within a margin of ± 0.05, regardless of the draft model chosen. Thus, we set our MCSD configuration starting with 2; through grid search in Figure 4, we find (2,4,3,1,1) yields the optimal speedup under our experiment environment on Mt-Bench. Moreover, when the number of target-initialized tokens is equal to two, the greedy sample does not yield a significant improvement in the acceptance rate and ends up with a speedup result similar to the baseline. Therefore, we are not presenting the result with the greedy sample.

In addition, we also set dynamic configuration with $D = 5$ and $W = 16$, where W (Width) indicates the number of candidate sequences generated in parallel at each step of the token tree, and D (Depth) represents the level of token prediction, referring to γ in traditional SD. The fixed $D = 5$ ensures the draft generation length is consistent across all SD methods. At the same time, $W = 16$ aligns the number of candidate sequences with those in MCSD, enabling a fair comparison between the methods.

Due to the quality of the generation, all settings in the following test follow the optimal MCSD static configuration. We will illustrate the ideal generation speed and corresponding acceptance rate based on the dynamic configuration in the Appendix section A.1.

4.2 Overall Results

The experiment result shown in Figure 3 demonstrates substantial improvements in generation speed. All of the configurations are shown and explained in Section 4.1. Conducted on the MT-Bench dataset, our method achieves a maximum speedup of **1.90** times over the baseline. The primary reason that using LlaMa-68M as the draft model results in greater speedup compared to TinyLlama-1.1B is that LlaMa-68M achieves approximately $5.5\times$ faster inference speeds than TinyLlama-1.1B. In

contrast, TinyLlama-1.1B shows only a $3.4\times$ higher acceptance rate with Llama-2-7B compared to the acceptance rate of LlaMa-68M.

Target Initialized Width	TinyLlama-1.1B	Llama-68M
2	5.07 (-1.22)	5.83 (-0.43)
3	4.32 (-1.97)	5.69 (-0.6)

Table 1: This shows the impact of target-initialized multi-candidate selection on generation quality (MT-Bench score) across various draft models, with a generation temperature of 0.7. The MT-Bench score (higher the better) for Llama-2-7B at this temperature is 6.29.

Dataset	Methods	Configuration	Llama 2-7B Chat	Vicuna-7B
MT-Bench	Baseline SD	$\gamma = 4$	0.17	0.18
	MCSD	$4 \times 2 \times 2 \times 1$	0.29	0.31
	Our method	$2 \times 4 \times 3 \times 1 \times 1$	**0.47**	**0.40**
TriviaQA	Baseline SD	$\gamma = 4$	0.15	0.21
	MCSD	$4 \times 2 \times 2 \times 1$	0.21	0.33
	Our method	$2 \times 4 \times 3 \times 1 \times 1$	**0.53**	**0.51**
Alpaca	Baseline SD	$\gamma = 4$	0.20	0.21
	MCSD	$4 \times 2 \times 2 \times 1$	0.34	0.35
	Our method	$2 \times 4 \times 3 \times 1 \times 1$	**0.52**	**0.46**

Table 2: Comparison of acceptance rate (α) for different methods using Llama-68M as draft model with temperature = 0.7 under MCSD static configuration

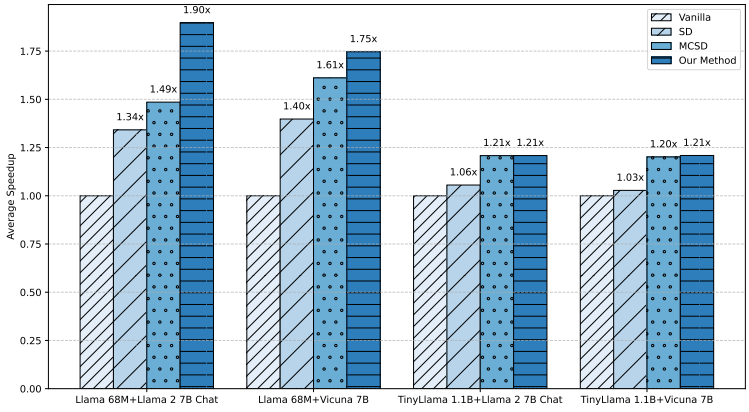

Figure 3: Speedup ratios compared to vanilla inference for different SD methods based on all three datasets under temperature = 0.7. We employ the static tree configuration. The bars represent speedup ratios for different model combinations (draft model + target model)

5 Discussion

We acknowledge that the combination of target-initialized multi-candidate generation, dynamic sliced topology-aware causal masking, and early stopping with a decision tree does not outperform static target model-initialized multi-candidate generation alone in terms of inference speed while preserving the highest generation quality. Two main factors contribute to this outcome:

1. In the dynamic MCSD generation, we employ a fork-shaped token tree. Since the highest generation quality is maintained when the number of target model-initialized tokens is set

to two, the fork-shaped token tree produces only two candidate sequences. This results in a lower acceptance rate compared to the tree-shaped token tree used in static target model-initialized multi-candidate generation.

2. The decision models we trained do not yield a substantial speedup via dynamic γ (shown in Appendix A.2) . Experimental results indicate that although early stopping provided by the decision model reduces draft generation time by avoiding likely rejected sequences, it also prematurely stops sequences that could validly extend for a longer acceptance length, increasing target model inference time. Consequently, the combined overhead from the decision model inference time and the extended target model inference time offsets the time saved in draft generation, leading to negligible speedup in most cases. Figure 7 suggests why our decision model leads to premature stopping.

Although we could address the first issue by sampling multiple draft tokens for each target model-initialized token and constructing the fork-shaped token tree based on these expanded draft tokens, this adjustment would not yield significant benefits without a decision model capable of accelerating MCSD inference.

We hope our work offers insights for future improvements in the speculative decoding process. For instance, while the dynamic MCSD process could theoretically enhance inference speed, our experimental results suggest that training an external decision model to perform early stopping—even if it could perfectly avoid premature stops and introduce no additional overhead to target model inference—would result in an overall speedup of no more than 10% compared to an optimal static MCSD process. Furthermore, MT-Bench results for target model-initialized multi-candidate generation suggest that when more than one target model-initialized token is used, a draft model with a higher acceptance rate relative to the target model preserves better target output quality than one with a lower acceptance rate. For instance, in a basic speculative decoding setup, TinyLlama-1.1B achieves an acceptance rate approximately 25% higher with Llama-2-7B compared to Llama-68M.

6 Conclusions

In this work, we propose a target-initialized multi-candidate token tree that enhances the acceptance rate in multi-candidate speculative decoding, with the output quality loss influenced by the number of target-initialized tokens and the specific draft model used. Additionally, we introduced a dynamic mask-slicing technique to avoid topology-aware causal mask generation overhead for dynamic multi-candidate speculative decoding. While we have yet to discover the decision model that could make dynamic multi-candidate speculative decoding faster than static one under all scenarios, our work can help future research find a more efficient multi-candidate speculative decoding process.

A Appendix

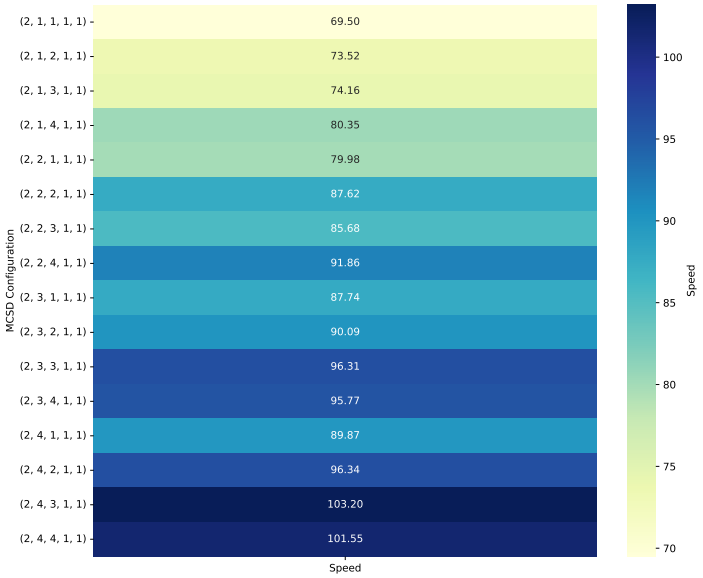

Figure 4: Speed heatmap across various static MCSD configuration

A.1 Dynamic MCSD Configurations and Results

This section thoroughly explores the results and configurations tested in our experiments with dynamic MCSD configuration. This section delves into the impact of various parameter settings on performance metrics, such as speed and acceptance rate, across different model configurations. Specifically, we analyze the effects of dynamic depth and width adjustments and acceptance rates achieved under different draft and target model combinations. Through these insights, this section aims to demonstrate our optimal outcomes when ignoring the quality lost.

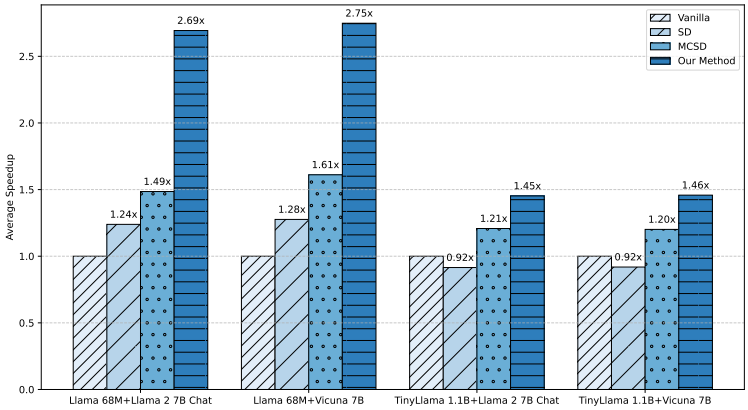

Figure 5: Speedup ratios compared to vanilla inference for different SD methods based on all three datasets under dynamic setting and temperature = 1. We employ dynamic tree configuration with $D = 5$ and $W = 16$. This is our optimal configuration based on empirical studies.

A.1.1 Effects of Width

We measure the effect of width on acceptance rate and speed in Figure 6 with fixed depth $D = 5$ based on empirical results. As the width increases, we consistently observe an improvement in

Dataset	Methods	Configuration	Llama 2-7B Chat	Vicuna-7B
MT-Bench	Baseline SD	$\gamma = 5$	0.11	0.13
	MCSD	$4 \times 2 \times 2 \times 1 \times 1$	0.20	0.23
	Our method	$D = 5, W = 16$	**0.74**	**0.75**
Alpaca	Baseline SD	$\gamma = 5$	0.13	0.13
	MCSD	$4 \times 2 \times 2 \times 1 \times 1$	0.23	0.23
	Our method	$D = 5, W = 16$	**0.75**	**0.79**
TriviaQA	Baseline SD	$\gamma = 5$	0.12	0.14
	MCSD	$4 \times 2 \times 2 \times 1 \times 1$	0.21	0.21
	Our method	$D = 5, W = 16$	**0.75**	**0.80**

Table 3: Comparison of acceptance rate (α) for different methods using Llama-68M as draft model with temp = 1. Since $D = 5$, we set $\gamma = 5$ and keep the original optimal setting of MCSD with a maximum tree length of 5

Dataset	Draft models	Temp	Acceptance Rate α	
			Llama 2-7B Chat	Vicuna-7B
MT-Bench	Llama-68M	0	0.76	0.77
	TinyLlama-1.1B	0	0.93	0.93
	Llama-68M	1	0.74	0.75
	TinyLlama-1.1B	1	0.95	0.95
Alpaca	Llama-68M	0	0.80	0.83
	TinyLlama-1.1B	0	0.92	0.94
	Llama-68M	1	0.75	0.79
	TinyLlama-1.1B	1	0.93	0.95
TriviaQA	Llama-68M	0	0.86	0.91
	TinyLlama-1.1B	0	0.96	0.94
	Llama-68M	1	0.75	0.80
	TinyLlama-1.1B	1	0.93	0.94

Table 4: Acceptance rates of our methods (temp = 0 and 1, given generation depth $D = 5$ and width $W = 16$)

acceptance rates and speed across different model pairs. The acceptance rate curve converges when $W = 12$, and the speed curve converges in $W = 10$. These findings demonstrate our method's effectiveness in improving acceptance rates and speed with relatively small tree widths. In addition, we notice there is a noticeable drop in speed when $W = 21$. This speed decline is due to the increased computational overhead associated with processing a more significant number of candidate tokens in parallel, which begins to outweigh the benefits of speculative decoding.

A.1.2 Acceptance Rate

We run acceptance rate α test across all three datasets with two different temperatures, shown in Table 4. Table 3 shows our method consistently demonstrating higher acceptance rate α across various datasets compared to baseline SD and MCSD. This indicates that our method's dynamic depth adjustment and target model initialization significantly enhance its ability to generate sequences more aligned with the target model's expectations. This leads to a higher overall acceptance rate, even under the same draft generation constraints.

A.2 Decision Model

To show the effect of the decision model in our framework, we compare the inference speeds under two scenarios, using or not using the decision model with different parameter size draft models. We

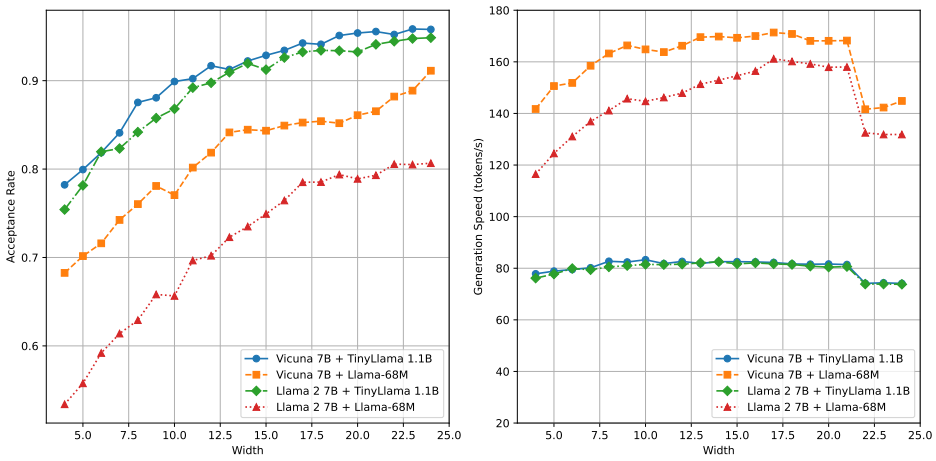

Figure 6: Left graph shows the relationship between width and acceptance rate (α) and right graph shows the the relationship between width and generation speed in tokens/s under MCSD dynamic configuration

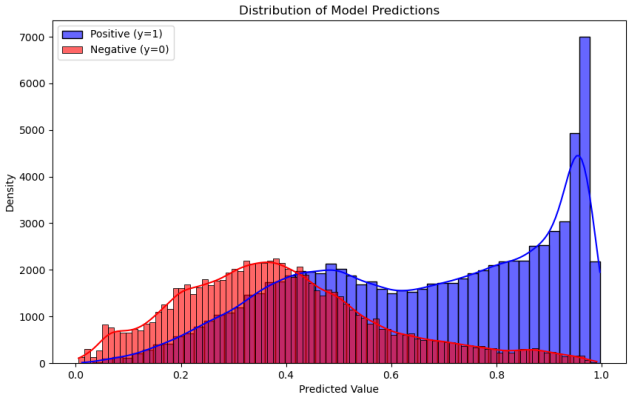

Figure 7: This figure demonstrates that, when using hidden states as input with labels of 0 or 1, the three-layer neural network struggles to effectively distinguish between negative and positive examples. As a result, a significant area of overlap is observed in the predictions.

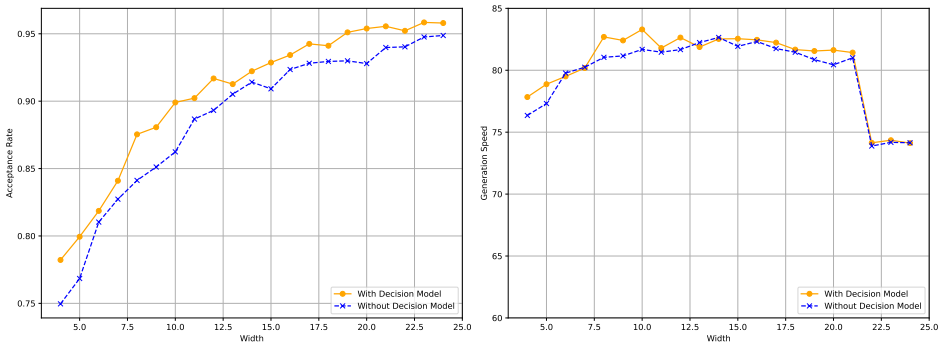

Figure 8: The left figure compares the acceptance rate, whether or not the decision is used, and the right figure compares its speed.

still use MT-bench as input prompts, and all other settings remain identical. As a result, in Figure 8,

both large and small draft models return higher speeds without using the decision model most of the time. The decision model only improves the speed slightly when the width is low. Nonetheless, in either case, the speed improvement of the decision model is minimal. In addition, it proves that our method's speedup is mainly due to the target model's initialized multi-candidate token tree and topology-aware causal mask rather than the decision model.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[3] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[4] Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[5] Y. Li, F. Wei, C. Zhang, and H. Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024.

[6] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, Z. Zhang, R. Y. Y. Wong, A. Zhu, L. Yang, X. Shi, C. Shi, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949, New York, NY, USA, 2024. Association for Computing Machinery.

[7] A. P. S, P. A. Nair, Y. Samaga, T. Boyd, S. Kumar, P. Jain, and P. Netrapalli. Tandem transformers for inference efficient llms, 2024.

[8] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[9] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[10] S. Yang, S. Huang, X. Dai, and J. Chen. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*, 2024.

[11] P. Zhang, G. Zeng, T. Wang, and W. Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

[12] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.