OPT-Tree: Speculative Decoding with Adaptive Draft Tree Structure

Jikai Wang[1], Yi Su[1]*, Juntao Li[1]†, Qingrong Xia[2], Zi Ye[2], Xinyu Duan[2], Zhefeng Wang[2], Min Zhang[1]
[1]Soochow University, China [2]Huawei Cloud, China
risus254@gmail.com, yisunlp@outlook.com, ljt@suda.edu.cn,
{xiaqingrong,yezi3,duanxinyu,wangzhefeng}@huawei.com,
zhangminmt@hotmail.com

Abstract

Autoregressive language models demonstrate excellent performance in various scenarios. However, the inference efficiency is limited by its one-step-one-word generation mode, which has become a pressing problem recently as the models become increasingly larger. Speculative decoding employs a ''draft and then verify'' mechanism to allow multiple tokens to be generated in one step, realizing lossless acceleration. Existing methods mainly adopt fixed heuristic draft structures, which do not adapt to different situations to maximize the acceptance length during verification. To alleviate this dilemma, we propose OPT-Tree, an algorithm to construct adaptive and scalable draft trees, which can be applied to any autoregressive draft model. It searches the optimal tree structure that maximizes the mathematical expectation of the acceptance length in each decoding step. Experimental results reveal that OPT-Tree outperforms the existing draft structures and achieves a speed-up ratio of up to 3.2 compared with autoregressive decoding. If the draft model is powerful enough and the node budget is sufficient, it can generate more than ten tokens in a single step. Our code is available at https://github.com/Jikai0Wang/OPT-Tree.

1 Introduction

Large language models (LLMs) (Black et al., 2022; Touvron et al., 2023; Jiang et al., 2024; Zheng et al., 2024) have achieved remarkable performance in various NLP scenarios. As models increase in size and complexity, the computational demands for inference rise significantly. Consequently, accelerating decoding is becoming increasingly important to save computing resources and reduce response time.

Autoregressive models (Black et al., 2022; Zhang et al., 2022; Touvron et al., 2023) usually generate one token in one decoding step, leading to limited decoding efficiency. In recent work, speculative decoding (Stern et al., 2018; He et al., 2023; Yang et al., 2023; Fu et al., 2024; Cai et al., 2024; Li et al., 2024) has shown great potential for lossless accelerated decoding. It applies a ''draft and then verify'' mechanism to maintain the original output distribution of the target model to be accelerated. Drafting is performed by a less-overhead drafting model. The generated draft is verified in parallel by the target model to generate multiple tokens in one decoding step, bringing promising acceleration.

Existing work like EAGLE (Li et al., 2024) has proposed methods for training small but effective draft models. Previous work mainly adopts drafts with structures of sequences or fixed trees. However, we argue that neither of them is the optimal draft structure under a limited node budget. Sequence-structured drafts (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2023; Yang et al., 2023; Zhang et al., 2023; Fu et al., 2024) contain redundant nodes. For example, ''A-B-C-D-E'' and ''A-B-C-F-G'' have the same prefix ''A-B-C'', which is calculated twice during verification. Therefore, there are only 7 valid tokens among the 10 nodes of these two sequences. Drafts with tree structure (He et al., 2023; Cai et al., 2024; Li et al., 2024; Jeon et al., 2024; Chen et al., 2024) solved this problem. The same token can appear only once in the same tree layer. A corresponding tree attention mask is designed for parallel verification. The specific structure of the tree is usually heuristic and remains constant. However, given a node budget, the best structure that maximizes the acceptance length during verification would change according to different inputs in each decoding step.

* Equal contribution.
† Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 13, pp. 188–199, 2025. https://doi.org/10.1162/tacl_a_00735
Action Editor: Ivan Titov. Submission batch: 6/2024; Revision batch: 11/2024; Published 2/2025.
© 2025 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

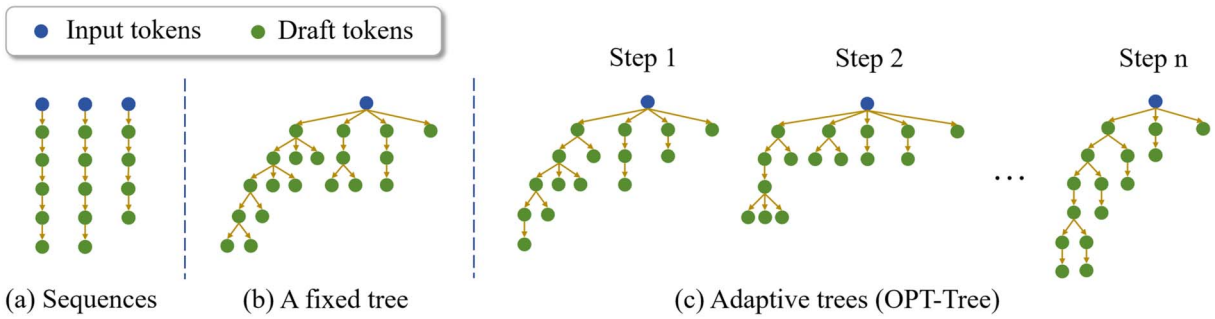

Figure 1: Draft structures used in speculative decoding. The blue node is the last token of the current input. The green nodes are tokens generated by the draft model. Nodes in the same layer share the same position index. OPT-Tree varies in each decoding step to achieve a larger acceptance length.

This paper proposes an adaptive and scalable tree structure called OPT-Tree. It can be applied to any autoregressive draft model. As is shown in Figure 1, the tree structure adaptively changes in each decoding step to maximize the mathematical expectation of the acceptance length. We apply a greedy algorithm to construct an OPT-Tree in each step. Details are elaborated in Section 3. We conduct comprehensive experiments in Section 4 to evaluate the effectiveness of OPT-Tree. Experimental results demonstrate that OPT-Tree outperforms the baselines and can be up to 3.2 times faster than vanilla autoregressive decoding. The mathematical expectation of the acceptance length is generally positively correlated with the actual acceptance length in practice. Moreover, OPT-Tree performs well when the tree size scales up. Using LLaMA-2-7B as the draft model, LLaMA-2-70B can generate 10 tokens in a single decoding step with OPT-Tree when the number of nodes is over 500, which indicates its great potential for adapting to more powerful computation resources and more effective draft models in the future.

2 Preliminaries

In this section, we provide the necessary definitions to establish a clear and precise foundation for the concepts discussed in this paper.

Inference. After inputting $\mathbf{x} = (x_1, x_2, \ldots, x_l)$, where l is the current sequence length, the target model M and the drafting model M_d return the next word distribution $p(y^{l+1}|x_1, x_2, \ldots, x_l)$ and $p_d(\hat{y}^{l+1}|x_1, x_2, \ldots, x_l)$, respectively, where y^{l+1} and $\hat{y}^{l+1}$ are the sampled next words.

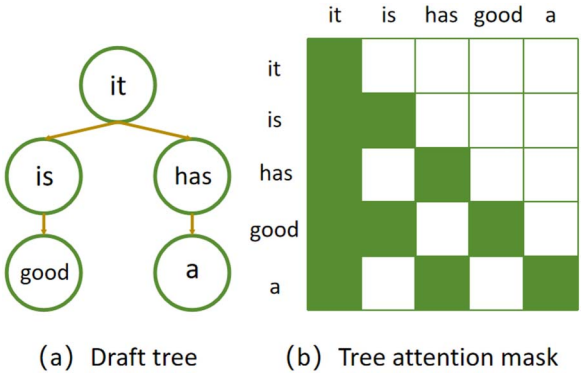

Figure 2: Subfigure (a) shows a draft tree, and subfigure (b) is its corresponding tree attention mask. The value of the blank position is zero.

Speculative Decoding. In speculative decoding with tree-structured draft, M_d first infers d steps to generate a draft tree T of depth d and then M verifies the draft. The verification depends on the sampling method. For greedy sampling, the ground truth is the sequence of tokens with the highest probability for each position output by M. As is shown in Figure 2, the model can process the tree structure input in parallel by constructing the corresponding tree attention mask. For all branches in the tree that contain the root node, the longest branch with the same prefix as the ground truth is accepted. Therefore, multiple tokens can be generated in one decoding step while ensuring that the generated sequences are consistent with the original ones. Due to the parallel computing mechanism, with given computing resources, the time cost of verification can be considered constant when the length of the verification sequence is within a certain range. As is shown in Figure 3, on a 4090 GPU with a model containing 7B parameters, the time required to verify a sequence of

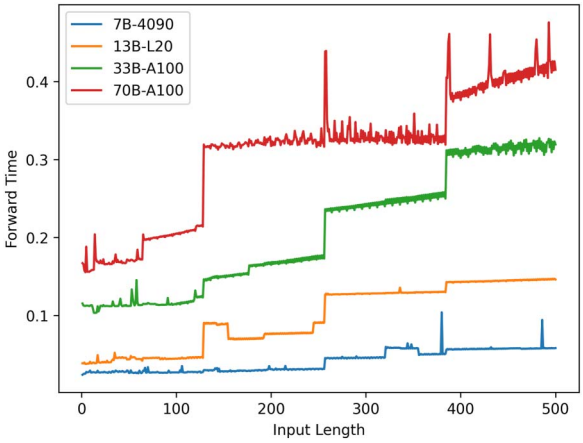

Figure 3: The relationship between input length and the wall time for inference for models of different sizes on various GPUs.

length 10 is similar to that for a sequence of length 200. Consequently, when discussing the construction of a draft tree in Section 3, we consider the node budget of the draft tree as a given condition.

3 OPT-Tree

This section introduces OPT-Tree, an algorithm for constructing our defined optimal draft tree structure for any input sequence in speculative decoding with autoregressive draft models.

Considering a certain step in speculative decoding whose input is **x**, the draft model M_d generates a draft tree based on **x** and the given tree structure T. Draft tree T is defined as follows:

$$\begin{aligned} T &= (\mathbb{V}, \mathbb{E}) \\ \mathbb{V} &= \bigcup_{i=l+1}^{l+d} \bigcup_{j=1}^{n_i} \left\{ (\hat{y}_j^i, p_d(\hat{y}_j^i)) \right\}, \end{aligned} \tag{1}$$

where $\mathbb{V}$ and $\mathbb{E}$ is the set of all nodes and edges. n_i represents the number of sampled tokens in the i_{th} layer of T. $p_d(\hat{y}_j^i)$ is the output probability of token $\hat{y}_j^i$ in M_d. For each node in T, if it has k children, they are k tokens greedily sampled according to p_d from its subsequent token distribution. Then the target model inputs the draft tree and the corresponding tree attention mask and returns the next tokens of each token in T. The root node is the last token of the current prompt, which is bound to be accepted. We get the longest accepted candidate with length A by comparing the next tokens and the draft tree. For the case where all leaf nodes are rejected, the acceptance length is 1.

Given M, M_d and n, for input **x**, an optimal tree structure T_{opt} should maximize the mathematical expectation of the acceptance length. Note that T_{opt} changes as the input changes. Since the optimization goal of the draft model is to make its output distribution close to the target model distribution, it is intuitive that the token with a larger output probability in draft model is more likely to be accepted. See Appendix A for verification of this property. Based on this, we can use p_d as the probability that it will be accepted to approximately calculate the mathematical expectation of the acceptance length. We use $E(A)$ to denote the approximation of the expected average acceptance length, which can be calculated by:

$$E(A) = \sum_{(\hat{y}_j^i, \hat{p}_j^i) \in T} \prod_{\hat{y} \in \mathbb{P}(\hat{y}_j^i)} p_d(\hat{y}), \tag{2}$$

where $\mathbb{P}(\hat{y}_j^i)$ is the set of all parent nodes of $\hat{y}_j^i$ (including itself). Note that the root node is also considered when calculating $E(A)$.

The process of solving T_{opt} is to find a subtree T of n nodes with the largest $E(A)$ from a complete n-ary tree. If there are no other conditions, the time complexity of solving this tree is typically $\Omega(n^2)$, which is unacceptable. However, by leveraging certain properties of the draft tree, we can design a more efficient algorithm to solve this problem.

For simplicity, we define the probability of the prefix as $\hat{p}$:

$$\hat{p}_j^i = \prod_{\hat{y} \in \mathbb{P}(\hat{y}_j^i)} p_d(\hat{y}). \tag{3}$$

$\hat{p}_j^i$ of the root node is regarded as 1. Then we can simplify the calculation of $E(A)$ as the summation of the probability of the prefix of all nodes according to the drafting model:

$$E(A) = \sum_{(\hat{y}_j^i, \hat{p}_j^i) \in T} \hat{p}_j^i. \tag{4}$$

Figure 4 shows a simple example of calculating $\hat{p}$ and $E(A)$. $E(A)$ should positively correlate with the acceptance length. We discuss their correlation in Section 4.2.

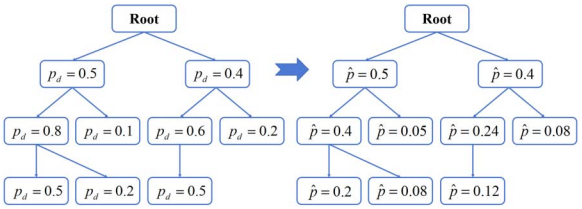

Figure 4: An example of a draft tree containing $\hat{p}$ in each node. p_d and $\hat{p}$ of the root are regarded as 1. The value of $E(A)$ is 3.07. See Appendix B for specific calculation process.

Algorithm 1 Construct an OPT-Tree T_{opt}

Input: Input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_l)$, draft model M_d, number of nodes n, threshold δ.

Output: A draft tree T_{opt}.

Initialize a tree T with root node x_l
$E \leftarrow 0$
Output distribution $P_d(T) \leftarrow M_d(T)$
$T \leftarrow topk(P_d(T), n)$
while Depth of tree $D(T) < n$ and $E_{sub}(T, n) - E > \delta$ **do**
  // Drafting step
  $E \leftarrow E_{sub}(T, n)$
  Output distribution $P_d(T) \leftarrow M_d(T)$
  $T \leftarrow topk(P_d(T), n)$
end while
$T_{opt} \leftarrow$ Select the n nodes with the largest $\hat{p}$ from T

We use $E_{sub}(T, n)$ to represent the maximum value of $E(A)$ for all subtrees of T that contain the root node and have n nodes. Note that the root node is not considered when calculating node trees and mathematical expectations.

Then, we propose Algorithm 1 to construct T_{opt} during the drafting phase for each decoding step. We initialize T with a root node. At each drafting step, we greedily sample n tokens with the largest $\hat{p}$ in the next token distributions of nodes in the last layer of T to construct the next layer. T has $d * n$ nodes at this time. Finally, we select the n nodes in T with the largest $\hat{p}$. It is easy to prove that these n nodes are a subtree of T, which contains the root node:

Proof. (1) If these nodes can not form a tree with the root, there is at least one node v_i whose parent node v_j is not among these nodes. (2) Since v_j is the parent node of v_i, $\hat{p}$ of v_j is larger than $\hat{p}$ of v_i. Therefore, v_j is also selected. (1) and (2) are contradictory, so these nodes must be able to form a subtree of T containing the root node. □

We summarize this as Theorem 3.1.

Theorem 3.1. *Given the depth of the tree, the top n nodes of the complete n-ary draft tree with the largest $\hat{p}$ form the T_{opt}.*

Theorem 3.2. *As the drafting step increases, $E_{sub}(T, n)$ is monotonic non-decreasing.*

According to Theorem 3.2, we can get the desired T_{opt} in theory by stopping drafting when $E(T)$ no longer increases. However, the draft model brings additional overhead to the practice. For autoregressive draft models, the drafting overhead is proportional to the depth of the draft tree.

Taking this into consideration, we introduce a threshold δ when setting the conditions for terminating drafting. The value of δ should be controlled between μ and 1, where μ is the time taken for one drafting step divided by the time taken for one decoding step.

A complete decoding step of M is detailed in Algorithm 2. In practice, both M and M_d utilize key and value cache (Pope et al., 2023) to calculate attention. This approach ensures that the actual input length of each drafting step is n, effectively preventing computational bottlenecks during the inference process of the draft model, even when operating under larger tree size budgets.

Algorithm 2 Speculative Decoding with Adaptive Draft Tree Structure

Input: Input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_l)$, target model M, draft model M_d, number of nodes n, threshold δ.

Output: New input sequence $\mathbf{x}' = (x_1, x_2, \ldots, x_{l+A})$

$T_{opt} \leftarrow$ Construct the draft tree with n nodes
$mask \leftarrow$ Compute the corresponding tree attention mask
$P \leftarrow M(T_{opt}, mask)$
$(y^{l+1}, y^{l+2}, \ldots, y^{l+A}) \leftarrow Verify(T_{opt}, P)$
// Find the longest accepted candidate. If a sequence of length $A-1$ successfully hits, its next word will also be accepted. So, the total acceptance length is A.
$\mathbf{x}' \leftarrow Concat(\mathbf{x}, (y^{l+1}, y^{l+2}, \ldots, y^{l+A}))$

M	M_d	Tree	MAL	Tokens/s	Speedup	M	M_d	Tree	MAL	Tokens/s	Speedup
LLaMA-2-7B	None	–	1.00	51.89	1.00	LLaMA-2-13B	None	–	1.00	26.79	1.00
	L-68M	Binary	2.12	68.58	1.32		L-68M	Binary	2.05	40.24	1.50
		EAGLE	2.47	77.06	1.49			EAGLE	2.42	46.82	1.75
		OPT-Tree	**2.58**	**87.57**	**1.69**			OPT-Tree	**2.58**	**48.10**	**1.80**
	L-1B	Binary	3.95	46.10	0.89		L-1B	Binary	3.95	37.37	1.39
		EAGLE	4.23	47.74	0.92			EAGLE	4.25	40.12	1.50
		OPT-Tree	**4.88**	**52.48**	**1.01**			OPT-Tree	**5.20**	**43.40**	**1.62**
	EAGLE	Binary	3.40	107.91	2.08		EAGLE	Binary	3.54	66.24	2.47
		EAGLE	3.73	130.50	2.51			EAGLE	3.80	73.97	2.76
		OPT-Tree	**4.36**	**132.75**	**2.56**			OPT-Tree	**4.35**	**76.61**	**2.86**
LLaMA-2-70B	None	–	1.00	6.29	1.00	Vicuna-33B	None	–	1.00	11.25	1.00
	L-7B	Binary	4.84	11.05	1.76		V-7B	Binary	4.41	12.49	1.11
		EAGLE	4.97	11.35	1.80			EAGLE	4.64	12.99	1.15
		OPT-Tree	**7.74**	**11.65**	**1.85**			OPT-Tree	**6.51**	**13.74**	**1.22**
	EAGLE	Binary	3.39	17.02	2.71		EAGLE	Binary	2.35	21.13	1.88
		EAGLE	3.67	18.81	2.99			EAGLE	2.69	24.92	2.21
		OPT-Tree	**4.06**	**19.21**	**3.05**			OPT-Tree	**3.06**	**25.17**	**2.24**

Table 1: Experimental results on MT-Bench. M_d being None represents vanilla autoregressive decoding. ‘‘L’’ and ‘‘V’’ in M_d column represent ‘‘LLaMA-2’’ and ‘‘Vicuna’’. ‘‘MAL’’ indicates ‘‘Mean Acceptance Length’’. The best results are shown in bold.

Applying our proposed algorithm to solve T_{opt} incurs an acceptable time cost. Detailed time costs for each operation are presented in Section 4.4.

4 Experiments

4.1 Main Results

Setup. We adopt LLaMA-2-7B, LLaMA-2-13B, LLaMA-2-70B (Touvron et al., 2023), and Vicuna-33B (Zheng et al., 2024) as target models to verify the effectiveness of OPT-Tree. We use a single GeForce RTX 4090 GPU for LLaMA-2-7B, a single L20 GPU for LLaMA-2-13B and 4 A100-PCIE-40GB GPUs for LLaMA-2-70B and Vicuna-33B. We choose one or two smaller models in the same version as the draft model for each target model. Moreover, we adopt a corresponding EAGLE draft model for each target model. EAGLE (Li et al., 2024) is an effective speculation decoding method that trains additional autoregressive heads as draft models. It uses a well-designed heuristic draft tree structure with 25 nodes. In our experiments, we regard it as the EAGLE draft tree. EAGLE is certified by Xia et al. (2024) as the fastest speculative method in their experiments. For each target and draft model group, we perform speculative decoding with greedy sampling and compare OPT-Tree with the Binary tree and EAGLE tree. The temperature is set to zero.

We compare the average acceptance length and number of tokens generated per second decoding with different tree structures. The speedup ratio is calculated according to generation speed. The number of nodes needs to be controlled within a certain range to avoid excessive time consumption in the verification phase. It is treated as a hyperparameter chosen in $[25, 50, 60]$ to maximize the speedup ratio according to different target models and GPU resources except for the EAGLE tree. We conduct evaluation on MT-Bench (Zheng et al., 2024) and GSM8K (Cobbe et al., 2021).

Results. Experimental results are shown in Table 1 and Table 2. Note that using LLaMA-2-1B as the draft model can hardly speed up decoding when the target model is LLaMA-2-7B because the difference in inference time between the two models is too small. EAGLE draft models achieve strong performance with fewer parameters, thus providing better acceleration than the small models in the same series with the target models. OPT-Tree outperforms other tree structures in terms of mean acceptance length in each group of experiments, especially when the performance of the draft model is close to the target model (e.g., LLaMA-2-70B combined with L-7B and Vicuna-33B combined with Vicuna-7B), indicating its high upper limit. Since OPT-Trees are usually deeper than binary trees and EAGLE trees, they incur more overhead when drafting. Therefore, from the perspective of tokens per second, the improvement is not as significant as that from

M	M_d	Tree	MAL	Tokens/s	Speedup	M	M_d	Tree	MAL	Tokens/s	Speedup
LLaMA-2-7B	None	–	1.00	52.76	1.00	LLaMA-2-13B	None	–	1.00	27.10	1.00
	L-68M	Binary	2.20	73.49	1.39		L-68M	Binary	2.21	45.18	1.67
		EAGLE	2.63	85.62	1.62			EAGLE	2.60	52.83	1.95
		OPT-Tree	**2.78**	**96.43**	**1.83**			OPT-Tree	**2.81**	**53.54**	**1.98**
	L-1B	Binary	3.55	40.69	0.77		L-1B	Binary	3.76	36.54	1.35
		EAGLE	3.87	44.42	0.84			EAGLE	4.10	37.29	1.38
		OPT-Tree	**4.46**	**50.83**	**0.96**			OPT-Tree	**5.10**	**42.97**	**1.59**
	EAGLE	Binary	3.52	118.15	2.24		EAGLE	Binary	3.80	73.30	2.70
		EAGLE	3.83	137.41	2.60			EAGLE	4.06	80.47	2.97
		OPT-Tree	**4.68**	**140.55**	**2.66**			OPT-Tree	**5.03**	**80.94**	**2.99**
LLaMA-2-70B	None	–	1.00	6.38	1.00	Vicuna-33B	None	–	1.00	10.74	1.00
	L-7B	Binary	4.85	11.20	1.76		V-7B	Binary	4.95	13.15	1.22
		EAGLE	4.98	11.51	1.80			EAGLE	4.81	13.38	1.25
		OPT-Tree	**7.62**	**12.10**	**1.90**			OPT-Tree	**6.35**	**13.98**	**1.30**
	EAGLE	Binary	3.62	18.63	2.92		EAGLE	Binary	2.82	25.20	2.35
		EAGLE	3.91	20.42	3.20			EAGLE	3.15	28.37	2.64
		OPT-Tree	**4.55**	**20.50**	**3.21**			OPT-Tree	**3.47**	**28.76**	**2.68**

Table 2: Experimental results on GSM8K. M_d being None represents vanilla autoregressive decoding. ‘‘L’’ and ‘‘V’’ in M_d column represent ‘‘LLaMA-2’’ and ‘‘Vicuna’’. ‘‘MAL’’ indicates ‘‘Mean Acceptance Length’’. The best results are shown in bold.

the mean acceptance length. Tokens per second are affected by different hardware resources and random errors. In addition, some method-independent techniques can also be used to reduce computation time. For example, the unchanged part of the attention mask in the drafting phase can be initialized only once and called multiple times, thus saving the time of multiple initializations. In order to make a fairer comparison in our experiments, we avoid these tricks to be consistent with EAGLE's practice. Overall, OPT-Tree outperforms the baselines. It can be up to about 3.2 times faster than vanilla autoregressive decoding. The similar performance on both datasets verifies the robustness of the proposed method.

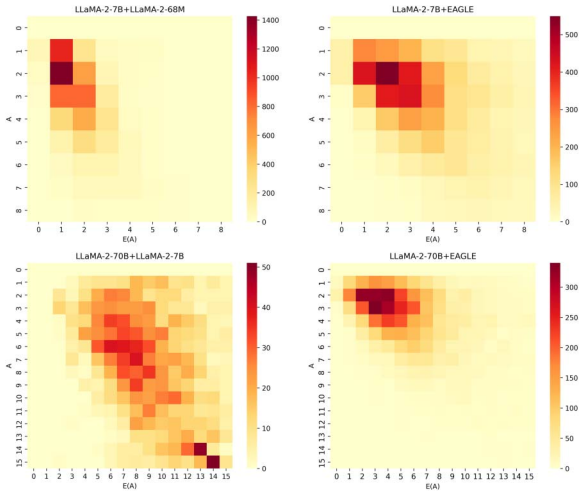

Figure 5: Correlation between $E(A)$ and A. The horizontal axis represents $E(A)$, and the vertical axis represents A. Each square shows the number of times the corresponding situation occurs. The darker the color, the more times it indicates.

4.2 Correlation between $E(A)$ and A

The theory of OPT-Tree is based on the premise that $E(A)$ is positively correlated with actual A. We record the values of $E(A)$ and A of OPT-Tree in about 8000 decoding steps for 4 groups of M and M_d. Figure 5 shows the results. The value of $E(A)$ is rounded. The darker areas in the four images are basically distributed along the main diagonal line. When $E(A)$ of the tree is larger, it also tends to get a more considerable acceptance length after verification. A stronger draft model shifts the distribution to the lower right corner. These phenomena corroborate our theoretical analysis. In addition, in the LLaMA-2-70B+LLaMA-2-7B group, high values of $E(A)$ and A (e.g., $E(A) = 14, A = 15$) are generally found, which demonstrates the potential of OPT-Tree to adapt to stronger draft models and larger draft tree sizes.

4.3 Scaling the Draft Tree Size

We conduct experiments to explore the changes in mean acceptance length with larger tree sizes. We compare OPT-Tree with Sequoia (Chen et al.,

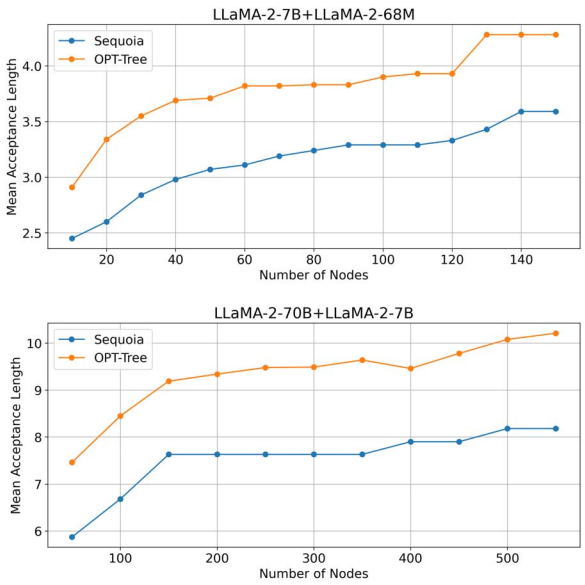

Figure 6: Mean acceptance length under different tree sizes under two sets of experiments.

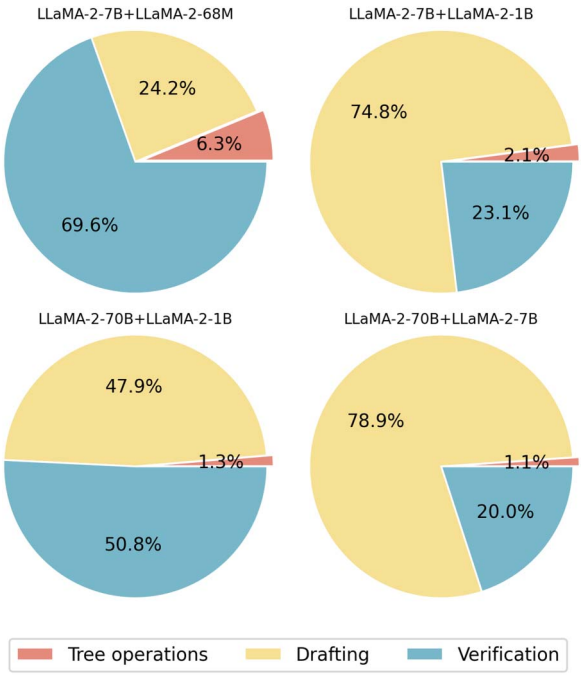

Figure 7: The average time cost of different operations in each decoding step. Tree operations include initializing the tree, updating the tree, calculating the optimal tree, and calculating the corresponding tree attention mask. The drafting time cost is the entire time to create a draft tree minus the time for tree operations. Its main component is the time of multiple draft model inferences. The verification time cost is mainly in the inference of the target model.

2024) using LLaMA-2-7B and LLaMA-2-70B as target models. Sequoia is a scalable draft tree that uses dynamic programming to solve for the tree structure. It requires the target and draft models to be used in advance to infer some samples to determine the best structure. The tree structure is fixed when doing speculative decoding. We use 200 samples in C4 (Raffel et al., 2020) to construct the Sequoia trees. Temperature is set to 0 in the experiments.

The results are shown in Figure 6. OPT-Tree outperforms Sequoia under various tree sizes. For LLaMA-2-7B+LLaMA-2-68M, the mean acceptance length increases when the number of nodes is smaller than 130 for both OPT-Tree and Sequoia. When the number of nodes exceeds 140, the mean acceptance length increases slowly. For LLaMA-2-70B+LLaMA-2-7B, the growth of mean acceptance length with Sequoia stabilizes when the number of nodes exceeds 150. In contrast, OPT-Tree can consistently improve the mean acceptance length even with more than 500 nodes. Since LLaMA-2-7B is a strong draft model for LLaMA-2-70B, the mean acceptance length can achieve 10 with an OPT-Tree of 500 nodes. A tree with 500 nodes costs a large amount of computation time for LLaMA-2-70B with A100-PCIE-40GB GPUs, thus being unable to speed up decoding in our practice. However, this cost may be acceptable if more powerful computational resources are equipped in the future.

4.4 Time Cost Analysis

In this section, we discuss the overhead of applying OPT-Tree. We conduct experiments with 4 groups of models on A100 GPUs. The number of nodes is set to 50. The temperature is set to 0.

Figure 7 displays the time cost of the operations in speculative decoding with OPT-Tree. Each pie in the figure represents the average wall time of one speculative decoding step, composed of tree-related operations time, drafting time, and verification time. Since the cost of tree-related operations is independent of the size of the draft and target models, these costs will constitute a smaller proportion when the models are larger. In the model setting with the least number of parameters, the time overhead of tree-related operations accounts for 6.3%. However, given the improvements in draft quality achieved by OPT-Tree, these costs are justified. When the model becomes sufficiently large (e.g., LLaMA-2-70B+LLaMA-2-7B), the time overhead associated with tree operations is negligible.

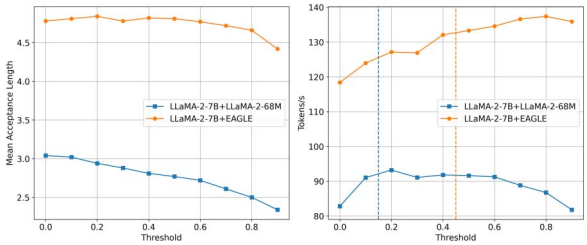

Figure 8: The mean acceptance length (length) and tokens/s (right) under different thresholds on MT-Bench. The target model is LLaMA-2-7B. The blue and orange dashed lines in the right figure represent the values of μ with LLaMA-2-68M and EAGLE as the draft model, respectively.

4.5 Impact of the Threshold

Considering that the overhead of the draft model is proportional to the depth of the tree, the tree that maximizes the acceptance length does not necessarily have the highest speed-up ratio. Therefore, we experiment to study the mean acceptance length and tokens/s under different thresholds.

Figure 8 shows the experimental results on LLaMA-2-7B. The mean acceptance length drops as the threshold grows when using LLaMA-2-68M as the draft model. However, there is a slight fluctuation for the EAGLE draft model. This is because $E(A)$ and A are not completely equivalent. We calculate μ for each group of models, which is the time of one drafting step divided by the time of one decoding step. A threshold that is too large will reduce the tree's depth, thus reducing the value of A. On the other hand, a threshold that is too small may make the tree too deep and increase the cost of drafting. When the depth of the tree increases by one but the increment of the $E(A)$ does not exceed μ, it is not worth increasing the depth. Therefore, we set a threshold between μ and 1 in practice. LLaMA-2-68M and EAGLE achieve the highest acceleration when $\delta = 0.2$ and $\delta = 0.8$, respectively.

4.6 Performance on Non-greedy Settings

In the decoding setting of non-greedy sampling (random sampling), we only modify the acceptable tokens during the verification phase. We evaluate OPT-Tree on these non-greedy settings, where the temperature exceeds 0.

We perform speculative decoding with OPT-Tree on the MT-Bench dataset for all groups of models in 4.1 with the temperature set to 1. Table 3 displays the experimental results. The mean acceptance length and the speedup ratio of speculative decoding with OPT-Tree are slightly lower when the temperature is set to 1 than when the temperature is set to 0. Since the draft tree greedily samples tokens with higher probability, the positive correlation between E(A) and A will be weakened in the decoding of random sampling. Therefore, it is typical for the acceleration of speculative decoding to drop when the temperature is greater than 0. The improvement of OPT-Tree for EAGLE draft model is more significant in this setting. Figure 9 shows specific changes in mean acceptance length and tokens/s with different temperature values. Both metrics drop as the temperature rises in general. But even when the temperature is set to 1, OPT-Tree can still provide high speedup compared to vanilla autoregressive decoding.

M	M_d	MAL	Tokens/s	Speedup
LLaMA-2-7B	L-68M	2.72	88.90	1.71
	L-1B	5.25	49.76	0.96
	†EAGLE	3.37	101.63	1.96
	EAGLE	4.07	125.79	2.42
LLaMA-2-13B	L-68M	2.26	43.45	1.62
	L-1B	4.23	37.84	1.41
	†EAGLE	3.45	63.13	2.01
	EAGLE	4.13	69.27	2.21
LLaMA-2-70B	L-7B	7.17	11.87	1.89
	†EAGLE	3.51	15.93	2.53
	EAGLE	4.09	18.92	3.01
Vicuna-33B	V-7B	4.91	13.48	1.20
	†EAGLE	2.70	19.91	1.77
	EAGLE	2.89	25.31	2.25

Table 3: Performance of OPT-Tree on MT-Bench with the temperature set to 1. ‘‘L’’ and ‘‘V’’ in M_d column represents ‘‘LLaMA-2’’ and ‘‘Vicuna’’. ‘‘MAL’’ indicates ‘‘Mean Acceptance Length’’. ‘‘†’’ means using the EAGLE tree.

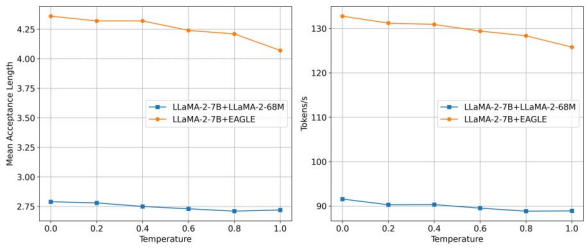

Figure 9: The two figures on the left and right are the mean acceptance length and tokens/s with OPT-Tree with different temperatures on MT-Bench. The target model is LLaMA-2-7B.

Figure 10: An example of speculative decoding with OPT-Tree on LLaMA-2-70B. Text on a blue background is the input prompt. Blue text represents drafts generated by LLaMA-2-7B and accepted by LLaMA-2-70B. Red text represents the next token for each accepted draft, which is generated by LLaMA-2-70B during the verification.

4.7 Case Study

We show an example of speculative decoding with an OPT-Tree of 50 nodes on LLaMA-2-70B with LLaMA-2-7B as the draft model in Figure 10. The threshold is 0.7, and the temperature is 0. The mean acceptance length is 9.34, and the generation speed is 12.07 tokens per second. Most words (blue text) are generated by the draft model and then verified by the target model. Each couple of red words and the continuous blue text in front of it is generated in a single decoding step of the target model. The appearance of red words is either because the depth of the draft tree is limited or because none of the candidates for this position hits the target. Prepositions (e.g., *in*, *for*, and *with*), conjunctions (e.g., *and* and *or*), articles (e.g., *a* and *the*), punctuation, and other words which have no apparent practical meanings in the drafts are prone to be rejected during the verification phase. In addition, the beginning of new sentences in drafts tends to be rejected because it has no solid sequential association with the previous word.

5 Related Work

Speculative decoding (Stern et al., 2018; Xia et al., 2023; Leviathan et al., 2023; Chen et al., 2023a) accelerates autoregressive decoding by drafting and then verifying while ensuring consistent output. Drafting methods are mainly divided into independent drafting and self-drafting. Independent drafting leverages an external low-cost model. SpecDec (Xia et al., 2023) trains a non-autoregressive model for drafting while others (Leviathan et al., 2023; Chen et al., 2023a; Spector and Re, 2023; Chen et al., 2023b, 2024) directly utilize a smaller version of the target model. In addition, REST (He et al., 2023) proposed a retrieval-based drafting method. Self-drafting uses the original information of the target model to draft. Yang et al. (2023) adopt an early-exiting mechanism for drafting. Similarly, Zhang et al. (2023) performs adaptive layer skipping in the drafting phase. Lookahead Decoding (Fu et al., 2024) designed an algorithm for parallel drafting and verification. MEDUSA (Cai et al., 2024) trains multiple decoding heads to obtain candidates for multiple steps from original features in parallel. Considering that different sampling results at each step in drafting will affect the distribution of subsequent output, EAGLE (Li et al., 2024) designed an autoregressive head, which introduced the embedding of each word in the drafting stage.

The verification method has evolved from sequence-structured verification to tree-structured verification. Early work (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2023; Yang et al., 2023; Zhang et al., 2023; Fu et al., 2024) verifies drafts in the form of one or several sequences. However, as the number of verification

tokens increases, there are a large number of prefix duplications between sequences, resulting in redundant calculations. To alleviate this problem, recent work (He et al., 2023; Cai et al., 2024; Li et al., 2024; Jeon et al., 2024) uses tree-structured heuristic drafts and designs the corresponding attention masks for parallel verification. Chen et al. (2024) proposed Sequoia, an algorithm for building draft trees, which performs well as the tree size scales up.

6 Conclusion

In this paper, we propose a novel and effective method called OPT-Tree to construct adaptive draft tree structures for speculative decoding, which is applicable to any autoregressive draft model. OPT-Tree maximizes the mathematical expectation of the acceptance length under any limited draft tree size. Experimental results across ten sets of target and draft models using two datasets demonstrate that OPT-Tree outperforms existing draft structures, achieving lossless acceleration of up to 3.2 times compared to vanilla autoregressive decoding. Furthermore, when paired with a robust draft model, OPT-Tree consistently increases the mean acceptance length even with over 500 nodes, showcasing its potential in scenarios with ample computational resources.

Acknowledgments

We want to thank all the anonymous reviewers and Action Editor Ivan Titov for their valuable comments. This work was supported by the National Science Foundation of China (NSFC No. 62206194 and 62276077), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), and Tecorigin.

References

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.bigscience-1.9

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774v3*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318v1*.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374v2*.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Jie Huang, and Kevin Chen-Chuan Chang. 2023b. Cascade speculative drafting for even faster LLM inference. *arXiv preprint arXiv:2312.11462v4*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168v2*.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using lookahead decoding. *arXiv preprint arXiv:2402.02057v1*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. 2023. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252v2*.

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. 2024. Recursive speculative decoding: Accelerating LLM inference via sampling without replacement. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

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.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. EAGLE: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077v2*.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Benjamin Spector and Chris Re. 2023. Accelerating LLM inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623v1*.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288v2*.

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. https://doi.org/10.18653/v1/2023.findings-emnlp.257

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851v3*.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. 2023. Predictive pipelined decoding: A compute-latency trade-off for exact LLM decoding. *arXiv preprint arXiv:2307.05908v2*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168v2*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open

pre-trained transformer language models. *arXiv preprint arXiv:2205.01068v4*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2024. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36.

A The Relationship between the Output Probability of the Draft Model and the Probability of Acceptance

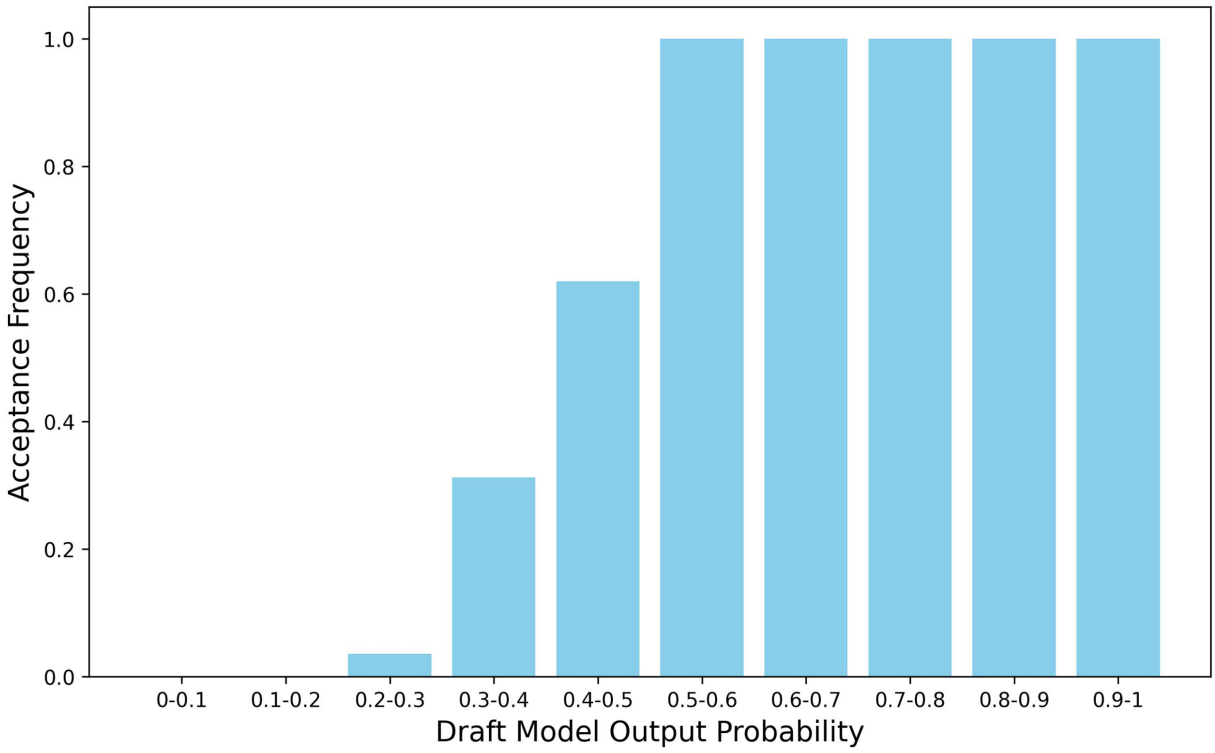

Figure 11: Acceptance frequency with different range of draft model output probability.

We counted the frequency of token acceptance under different draft model output probabilities when using LLaMA-2-7B and the EAGLE draft model for speculative decoding. Figure 11 shows the experimental results. The acceptance frequency is positively correlated with the output probability.

B Calculation Process in Figure 4

To calculate $E(A)$ by p_d (left tree):

$$\begin{aligned} E(A) = & \ 4 \times (0.5 \times 0.8 \times 0.5 + 0.2 \times 0.8 \times 0.5 + 0.5 \times 0.6 \times 0.4) \quad \text{—— Layer 3} \\ & + 3 \times [0.8 \times 0.5 \times (1 - 0.5 - 0.2) + 0.5 \times 0.1 \\ & + 0.6 \times 0.4 \times (1 - 0.5) + 0.4 \times 0.2] \quad \text{—— Layer 2} \\ & + 2 \times [0.5 \times (1 - 0.8 - 0.1) + 0.4 \times (1 - 0.6 - 0.2)] \quad \text{—— Layer 1} \\ & + 1 \times (1 - 0.5 - 0.4) \quad \text{—— Root} \\ = & \ 3.07 \end{aligned}$$

To calculate $E(A)$ by $\hat{p}$ (right tree):

$$\begin{aligned} E(A) &= 1 + 0.5 + 0.4 + 0.4 + 0.05 + 0.24 + 0.08 + 0.2 + 0.08 + 0.12 \\ &= 3.07 \end{aligned}$$