Graph-Structured Speculative Decoding

Zhuocheng Gong[1]*, Jiahao Liu[2], Ziyue Wang[3], Pengfei Wu[1]
Jingang Wang[2], Xunliang Cai[2], Dongyan Zhao[1,4]†, Rui Yan[5]†
[1]Wangxuan Institute of Computer Technology, Peking University
[2]Meituan; [3]Tianjin University; [4]National Key Laboratory of General Artificial Intelligence
[5]Gaoling School of Artificial Intelligence, Renmin University of China
{gzhch,zhaody}@pku.edu.cn, pengfeiwu1999@stu.pku.edu.cn
ruiyan@ruc.edu.cn, wangziyue@tju.edu.cn
{liujiahao12,wangjingang02,caixunliang}@meituan.com

Abstract

Speculative decoding has emerged as a promising technique to accelerate the inference of Large Language Models (LLMs) by employing a small language model to draft a hypothesis sequence, which is then validated by the LLM. The effectiveness of this approach heavily relies on the balance between performance and efficiency of the draft model. In our research, we focus on enhancing the proportion of draft tokens that are accepted to the final output by generating multiple hypotheses instead of just one. This allows the LLM more options to choose from and select the longest sequence that meets its standards. Our analysis reveals that hypotheses produced by the draft model share many common token sequences, suggesting a potential for optimizing computation. Leveraging this observation, we introduce an innovative approach utilizing a directed acyclic graph (DAG) to manage the drafted hypotheses. This structure enables us to efficiently predict and merge recurring token sequences, vastly reducing the computational demands of the draft model. We term this approach Graph-structured Speculative Decoding (GSD). We apply GSD across a range of LLMs, including a 70-billion parameter LLaMA-2 model, and observe a remarkable speedup of 1.73× to 1.96×, significantly surpassing standard speculative decoding[1].

1 Introduction

The impressive performance of Large Language Models (LLMs) comes with an efficiency bottleneck that hinders their broader adoption (Vaswani et al., 2017; Touvron et al., 2023a; OpenAI, 2022; Touvron et al., 2023b). In this context, speculative decoding (SD) emerges as a promising direction to accelerate the decoding process by reducing the number of forward passes of LLMs (Chen et al., 2023; Leviathan et al., 2023; Zhou et al., 2023; Spector and Ré, 2023; Miao et al., 2023a). The underlying idea of SD is “draft then verify": rather than generating one token at a time using the LLM, SD employs a smaller model to draft a hypothesis sequence of tokens covering several decoding steps and then uses the LLM to verify the hypothesis. Consequently, the decoding process includes a *draft stage* and a *verification stage*. In this scheme, the number of forward calls of LLMs can be significantly reduced.

Hypothesis 1
The hungry purple dinosaur ate the kind, zingy fox.
Hypothesis 2
The hungry purple dinosaur play with the kind, zingy fox.

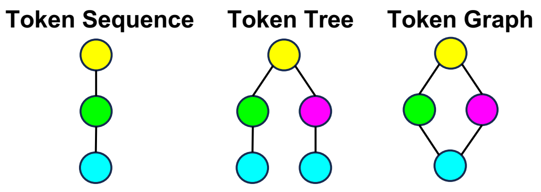

Figure 1: An illustrative comparison between the tree- and graph-structured draft token management.

However, SD faces its own set of challenges: the trade-off between performance and efficiency of the draft model limits the potential for acceleration. Ideally, the draft model should generate high-quality hypotheses while maintaining computational efficiency — a balance that is notoriously difficult to strike, echoing the adage that "there’s no such thing as a free lunch." In this study, we address

[1]Code available at https://github.com/gzhch/gsd
*Work done during an internship at Meituan.
†Corresponding authors: Dongyan Zhao (zhaody@pku.edu.cn) and Rui Yan (ruiyan@ruc.edu.cn).

the challenge of enhancing the acceptance rate of the draft model's hypotheses without increasing the computational burden. Inspired by beam search (Graves, 2012) and tree attention (Spector and Ré, 2023; Miao et al., 2023a), our approach involves producing a bunch of hypotheses instead of a solitary one. Then, the LLM verifies these multiple hypotheses in a singlar forward pass and accepts the longest one. While tree decoding, which adopts a tree structure to organize the drafted tokens, presents an efficient implementation for simultaneously drafting all hypotheses, it also leads to exponential growth in the number of tokens at deeper levels of the tree, resulting in a prohibitive computational overhead. Consequently, the length of the hypotheses must be kept relatively short, which in turn leads to suboptimal use of the draft model's capabilities.

Our objective is to extend the length of drafted hypotheses without a corresponding rise in computational cost. To this end, we meticulously examined the hypotheses to find opportunities for improvement. We observe that hypotheses based on the same context are often semantically similar or related, and the variations among differing hypotheses typically boil down to only a handful of tokens. Notably, more than 70% of the drafted tokens tend to recur across various hypotheses. If we could discern when the draft model is likely to predict these re-occurring tokens, we could simply reuse them from previous drafts, thereby reducing the overall number of tokens that need to be generated. Capitalizing on this revelation, we propose Graph-structured Speculative Decoding (GSD), which uses a directed acyclic graph to organize the drafted tokens (Figure 1). In this graph, each path that stems from the root node corresponds to a unique hypothesis. This approach allows different hypotheses to share a substantial number of common nodes.

The pipeline of GSD follows that of standard SD (also the Sequence-structured SD, SSD), which encompasses a draft stage and a verification stage. In the draft stage, the draft model constructs a token graph containing multiple hypotheses. In the verification stage, the token graph is flattened into a sequence, enabling the LLM to validate all hypotheses concurrently. The longest one is then adopted as part of the final output. We conduct extensive experiments using LLaMA-70b, one of the largest open-source LLMs, showing that GSD drafts tokens not exceeding $2\times$ the amount drafted by SSD on average, while tree-structured SD (TSD) drafted a token count that is more than 15 times greater. In terms of speedup, GSD outperforms all other methods, marking a significant advancement in speculative decoding techniques

2 Related Works

2.1 LLM Compression

Improving the efficiency of LLM inference has emerged as a pivotal research focus in recent years. The primary objective of model compression is to decrease computational demands and speed up the inference process. Research into the compression of large language models branches out into several directions, including knowledge distillation (Jiao et al., 2020; Sanh et al., 2019; Wang et al., 2021; Passban et al., 2021), quantization (Tao et al., 2022; Liu et al., 2023a,b; Dettmers et al., 2023; Xiao et al., 2023), network pruning (Liang et al., 2021; Frantar and Alistarh, 2023). Despite their innovations, these methods can be classified as lossy compression. This means that their efficiency improvements are intrinsically linked to a trade-off in performance, leading to the likelihood that a compressed LLM might produce compromised results.

2.2 LLM Decoding Acceleration

Alongside conventional model compression techniques, there is another branch of research that focuses on accelerating LLM inference without incurring information loss. Among these studies, speculative decoding (SD) (Chen et al., 2023; Leviathan et al., 2023; Zhou et al., 2023; Spector and Ré, 2023; Miao et al., 2023a) emerges as a promising technique. SD does not modify the model architecture, nor does it require supplemental data or retraining. SD typically employs a smaller model to draft initial predictions for “easy" tokens, while the LLM itself verifies these drafted tokens and generates “hard" tokens. Some researchers suggest that the smaller model is not essential for SD. For instance, the smaller model can be substituted with the LLM itself (Zhang et al., 2023) or a large text database (He et al., 2023). In addition to SD, other efforts are being made to enhance the decoding efficiency of LLMs. Blockwise parallel decoding (Stern et al., 2018), for example, is introduced to make predictions for multiple time steps in parallel. More recently, Medusa (Cai et al., 2023) has trained multiple prediction heads to predict the next set of tokens simultaneously.

3 Preliminaries: Sequence-structured Speculative Decoding

In this section, we establish the notation and provide a foundational overview of sequence-structured speculative decoding (SSD).

Consider an input sequence at time step t, denoted by $x_{\leq t} = \{x_1, x_2, ..., x_t\}$, where each x_i symbolizes the i-th token from the sequence. Let M_p be the target LLM we want to accelerate, and let M_q denote the draft model. The probabilities $p(x_{t+1}|x_{\leq t})$ and $q(x_{t+1}|x_{\leq t})$ represent the predictive distributions for the next token as given by M_p (the LLM) and M_q (the draft LM), respectively.

SSD leverages the draft model, M_q, to propose a hypothesis comprising γ tokens, which we denote as $h = \{\tilde{x}_{t+1}, \tilde{x}_{t+2}, ..., \tilde{x}_{t+\gamma}\}$. The drafting of each token, $\tilde{x}_t + i$, is modeled as follows:

$$\tilde{x}_{t+i} \sim q(x|x_{\leq t}, \tilde{x}_{t+1}, ..., \tilde{x}_{t+i-1}) \qquad (1)$$

Upon completion of the draft stage, the LLM verifies the γ drafted tokens in a singular forward pass. The verification process, which compares predictions made by M_p and M_q to determine which tokens shall be accepted, can be conducted in both deterministic and non-deterministic ways. Deterministic verification accepts drafted tokens only if the LLM would generate the same. The non-deterministic way employs the sampling method used in previous studies (Chen et al., 2023). For the i-th token in the hypothesis, the acceptance probability is calculated as $\min(1, p(\tilde{x}_{t+i})/q(\tilde{x}_{t+i}))$. Should the token $\tilde{x}_{t+i}$ face rejection, all subsequent tokens in the hypothesis are also discarded, the verification process comes to a halt, and M_p regenerates the discarded token. This method ensures that the tokens that are ultimately accepted are representative of the output distribution characterized by M_p.

4 A Step Forward: Tree-structured Speculative Decoding

An intuitive idea for improving SSD is to draft multiple hypotheses instead of merely one. This is where Tree-structured SD (TSD) comes into play.

In each drafting step of SSD, the draft model predicts a single next token as described in Equation 1. After γ steps, the drafted tokens compose a sequence $\{\tilde{x}_{t+1}, \tilde{x}_{t+2}, ..., \tilde{x}_{t+\gamma}\}$. In contrast, TSD allows the draft model to consider k different alternatives for the next token at each drafting step. The resulting drafted tokens thus create a tree structure, with the root representing the context at the commencement of drafting, and each branch from the root depicting a different hypothesis.

After γ drafting steps, the resulting token tree has a depth of γ and a maximum out-degree of k and can contain up to $\frac{k^{\gamma+1}-1}{k-1}$ nodes, representing as many as k^γ unique hypotheses. Let's denote the collection of all hypotheses as $\{h_i\}_{i=1}^{k^\gamma}$. TSD holds a significant advantage over SSD; by enabling the generation of a larger pool of hypotheses in a single drafting stage, it raises the chances of having longer sequences of tokens accepted by the LLM. This boosts the acceptance rate of the SD process. Fundamentally, TSD operates in a manner analogous to beam search, maintaining multiple potential hypotheses within its tree structure during the draft stage and then selecting the most promising one during the verification stage.

4.1 Parallelized drafting and verifying via tree attention

The draft stage of TSD generates a multitude of hypotheses. A significant challenge within this framework is the efficient drafting of these multiple hypotheses. If one were to adhere to the traditional inference scheme that decodes one token at a time (akin to extending one branch of the token tree), the computational demands are apparently unacceptable given that the token tree contains $\frac{k^{\gamma+1}-1}{k-1}$ tokens to be decoded.

A promising resolution to this problem is by employing meticulous tree attention. Tree attention operates by flattening the token tree into a sequence and then simultaneously predicting the next node for all branches during a single forward draft, thus circumventing the necessity of performing a forward pass for each potential sequence. As illustrated in Figure 2, it accomplishes this by customizing the attention mask in such a way that each token is only allowed to attend to its ancestor nodes in the tree hierarchy, thus maintaining the correct dependencies amongst tokens.

The verification stage benefits from tree attention by validating all hypotheses within a single forward pass. After this process, the longest path that unfolds from the root node is chosen as the sequence to be accepted.

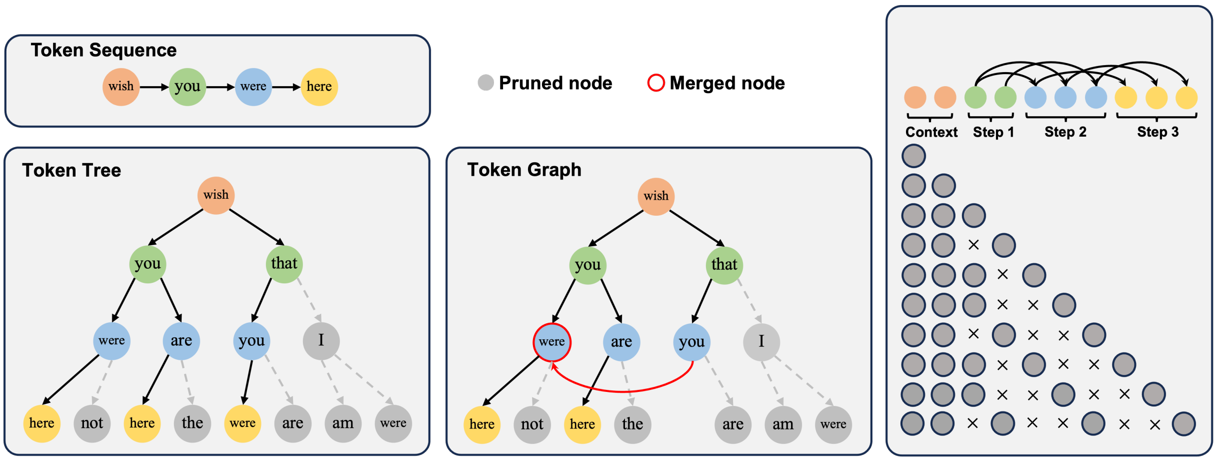

Figure 2: Overview of our method. (Left) GSD advances beyond TSD and SSD by implementing pruning strategies along with a re-occurring node merging technique. (Right) An illustration demonstrates the process by which the token tree (or graph) is flattened to a sequence. The sequence is then paired with a customized attention mask designed to uphold the proper dependencies between tokens to perform efficient drafting and verifying.

4.2 Pruning inferior branches

Despite the parallel drafting and verification with tree attention, TSD still consumes significantly more computation than SSD. The root cause lies in the exponentially increased length of input sequences processed in each forward pass. Transformer attention has a computational complexity that scales quadratically, $\mathcal{O}(l^2)$, with the sequence length l. While kv-caching does alleviate the computational load to some degree, the burden remains substantially heavier than that of SSD. Thus, to reduce the input sequence length, we need to perform pruning on the token tree.

We introduce two pruning strategies to moderate the size of the token tree. The first strategy is *probability pruning*. For a given node c within the token tree, where s_c denotes the path from the root to c, the logit probability is given by $q(c|x_{\leq t}, s_c)$. By setting a probability threshold θ_{prob}, we can filter out nodes: if $q(c|x_{\leq t}, s_c) < \theta_{prob}$, the node is deemed unlikely to be verified successfully and is marked as a leaf, halting further speculation.

The second strategy, *sibling pruning*, focuses on a node's child nodes $\{c_i\}_{i=1}^k$. Among these, we discern which nodes should remain as non-leaf nodes based on their logit probabilities relative to the highest probability among them. Specifically, let $m_q = \max_{i=1,\dots,k} p(c_i|x_{\leq t}, s_{c_i})$. A child node c_i is then designated as a leaf if $p(c_i|x_{\leq t}, s_{c_i}) < \theta_{sib} \cdot m_q$. This approach ensures that the logit probabilities among sibling nodes do not deviate excessively from the maximum observed, m_q. The underlying idea is that, during probabilistic sampling, if the generation probabilities across a node's children vary greatly, the tokens associated with lower probabilities are less likely to be chosen. Therefore, it may not be necessary to keep these less probable nodes in the tree. Hence, when the output distribution for a current token is peaked—indicating high model confidence in its prediction—we need not preserve many child nodes. However, if the distribution is flatter, meaning multiple tokens have similar probabilities, it then becomes prudent to maintain a broader set of child nodes as candidates.

5 Graph-structured Speculative Decoding

Empirically, we observe that TSD often fails to surpass SSD, contrary to expectations. It appears that despite the utilization of pruning and tree attention, the cost of drafting multiple hypotheses still counterbalances the potential benefits that TSD offers. So we would like to ask: Can we further reduce the quantity of drafted tokens to enhance TSD's efficiency and effectiveness?

5.1 Same tokens re-occur among hypotheses

Before delving into GSD, we first conduct a pilot study to investigate the drafted hypotheses generated by TSD. We analyze the token trees from 100 distinct TSD runs, documenting the statistics of n-gram co-occurrences across various branches. The findings of this analysis are presented in Figure 3, and they give rise to several key insights:

- There is a high degree of commonality among the tokens in different hypotheses. As depicted in Figure 3, within a token tree of 10-

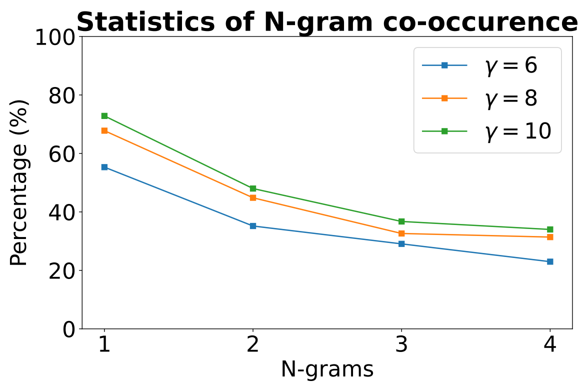

Figure 3: The proportion of tokens that are part of re-occurring n-grams within the token tree where the maximum out-degree k is 4. $\theta_{prob} = 0.2$ and $\theta_{sib} = 0.3$.

depth and 4-width, approximately 70% of tokens appear across multiple branches. This suggests that the generated hypotheses tend to form a cluster of semantically similar or related candidates, rather than branching off in completely disparate semantic directions.

- There is also a notable frequency of recurring n-grams within the token tree. This observation suggests that the similarities between different hypotheses extend beyond single tokens — entire segments of tokens (n-grams) are often duplicated among the various branches of the tree. This pattern points to redundancy in the token sequences being drafted, which may have implications for optimizing the efficiency of the speculative decoding process.

5.2 Identifying redundant nodes

We leverage the findings of identical tokens reappearing across different hypotheses to reduce computation.To this end, we introduce the concept of a τ-redundant node. A node is designated as τ-redundant when it corresponds to the last token of a re-occurring τ-gram. We assume that the presence of a τ-gram, defined as a sequence of τ consecutive identical tokens, signals a high degree of similarity between the current hypothesis and an alternate hypothesis already explored. This implies a strong likelihood that the sequence will continue to predict identical subsequent tokens.

5.3 Merging redundant nodes

Building on the concept of τ-redundant nodes, we implement a procedure to merge these nodes to enhance efficiency. The approach is straightforward: we mark τ-redundant nodes as leaf nodes, effectively ceasing their further expansion within the token tree. To merge the nodes, we first locate the first occurrence of the re-occurring τ-gram. We then draw a directed edge from the τ-redundant node to this first occurrence. By doing so, we establish that the nodes following the τ-redundant node will not need to be generated anew. Rather, we can directly reuse the results previously computed for the initial τ-gram occurrence. As a result of this merging process, the token tree is transformed into a directed acyclic graph (DAG), wherein no n-grams longer than τ will be repeated.

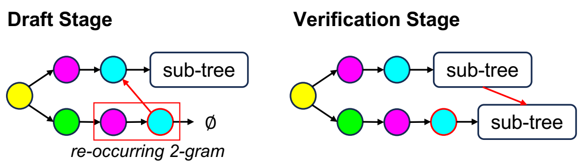

Figure 4: An illustration of how the token graph operates during the draft stage and the verification stage.

How does node merging hurt the performance? Merging nodes can result in a divergence from the nodes that would have otherwise been generated, potentially impacting the quality of the generated content. To quantify this effect, we calculate the KL divergence between the probability distributions of the next token across the vocabulary with or without node merging. Experimental results demonstrate that the KL divergence decreases rapidly with the increase of τ, suggesting that the impact of node merging diminishes significantly as the threshold τ is heightened. (Detailed results in Appendix C)

5.4 Token graph verification

There is still one step to go to fulfill GSD: the verification process. In the verification stage, we need to flatten the token graph to a sequence so that the LLM can verify all hypotheses simultaneously. To convert a DAG into a sequence while preserving the correct dependencies between tokens, we start by reverting the graph to its original tree structure. This is done by “unmerging" all previously merged nodes. During this process, the successor nodes of any redundant node are replicated from the relevant merged nodes (Figure 4). With the structure now back in the form of a tree, we can apply the same verification procedure as used in TSD.

Datasets	Mehtod	Model	Acceptance Rate	Drafted Token Num	Graph Success	Speedup
GSM8k	Self SSD	LLaMA-2-70b	-	-	-	1.37×
GSM8k	SSD	LLaMA-2-70b	0.795	629.9	-	1.85×
GSM8k	TSD	LLaMA-2-70b	0.894	8574.6	0%	1.81×
GSM8k	GSD	LLaMA-2-70b	0.917	793.1	27.7%	**1.96**×
XSUM	Self SSD	LLaMA-2-70b	-	-	-	1.28×
XSUM	SSD	LLaMA-2-70b	0.652	773.2	-	1.56×
XSUM	TSD	LLaMA-2-70b	0.784	22512.4	0%	1.42×
XSUM	GSD	LLaMA-2-70b	0.831	1544.8	32.8%	**1.73**×
XSUM	SSD	LLaMA-2-70b-chat	0.496	989.4	-	1.19×
XSUM	TSD	LLaMA-2-70b-chat	0.634	4601.2	0%	1.30×
XSUM	GSD	LLaMA-2-70b-chat	0.642	1545.7	30.4%	**1.32**×

Table 1: Evaluation results on 70b model. Self SSD is the method proposed by Zhang et al. (2023), which uses the LLM itself as the draft model. Speedup is the averaged result of greedy and top-p sampling. Here we only present the results of 70b models, full results can be found in Appendix D.

6 Experiments

6.1 Setup

There are two settings for verifying the drafted tokens: a deterministic setting where accepting the drafted tokens only if the LLM would generate tokens the same, and a non-deterministic setting where accepting the drafted tokens if they follow the same distribution with the LLM-itself generated tokens. In our main experiments, we adhere to the deterministic decoding setting if not specified. Under this condition, the generated output sequence is guaranteed to be identical to what would be produced via standard generation methods, so we can concentrate solely on efficiency metrics. Other details can be found in Appendix A.

Models We experiment on various backbone LLMs, including LLaMA (Touvron et al., 2023a), OPT (Zhang et al., 2022), and BLOOM (Workshop et al., 2022). For LLaMA, we use LLaMA-70b, LLaMA-70b-chat, and LLaMA-7b as large LLMs and LLaMA-7b and LLaMA-7b-chat, LLaMA-160m as draft models respectively. Note that LLaMA-160m is not an official checkpoint but a LLaMA-like model (Miao et al., 2023b). For OPT, we use OPT-13b as the LLM and OPT-350m as the draft model. For BLOOM, we use BLOOM-7b1 as the LLM and BLOOM-560m as the draft model.

Datasets We evaluate on Extreme Summarization (XSum) (Narayan et al., 2018), GSM8K (Cobbe et al., 2021), Alpaca (Taori et al., 2023), and WMT-14 (En-De) (Bojar et al., 2014). For GSM8K and WMT-14, we evaluate the full test set. For XSum and Alpaca, we randomly select 5000 instances for evaluation.

6.2 Main Results

Table 1 illustrates a comparison of our method against other speculative decoding approaches. Focusing on the speed-up ratio, we can see that GSD offers a significant advantage over the alternatives, achieving up to 1.94 and 1.70 times faster speeds. When examining the acceptance rate, we observe that both TSD and GSD have an acceptance rate that exceeds that of SSD by more than 10%. This indicates that tokens generated by the draft model are more likely to pass the verification process. Comparing the number of drafted tokens, we can see that TSD produces an order of magnitude more tokens than SSD. Hence, while TSD also has a high acceptance rate, this advantage is negated by the excessive number of tokens generated.

Additionally, we assess what proportion of tokens, which passed verification during the speculative decoding process, contained nodes from the merged subtrees, and find that approximately 30% of the drafting stages include such tokens. This indicates that, while the token graph is significantly smaller in node count compared to the token tree, we have successfully preserved the decoding information by recognizing and grafting nodes from different branches.

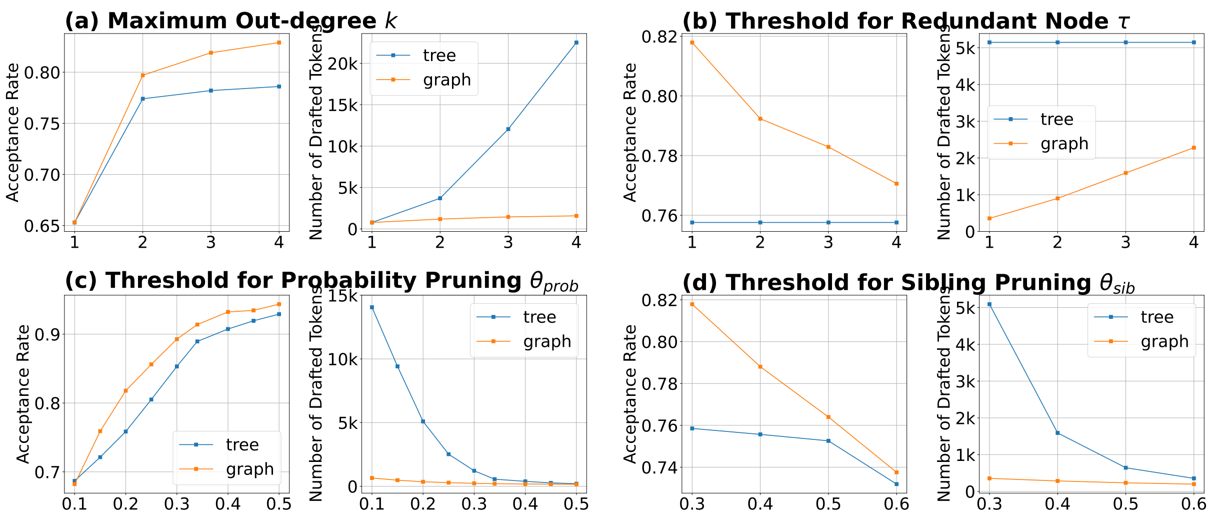

Figure 5: A series of ablation studies to investigate the hyperparameter configuration of maximum out-degree, redundant threshold, and two pruning techniques. All other hyperparameters adhere to the configuration described in section A.

Methods	SSD	TSD	GSD
GSM8k	1.80×	1.81×	2.14×
XSUM	1.58×	1.46×	1.89×

Table 2: Speedup results on non-deterministic speculative decoding on LLaMA-2-70b.

6.3 Ablation Study

Maximum Out-degree k Maximum out-degree k refers to the maximum number of child nodes that each node within the token tree (or graph) can possess. As depicted in Figure 5(a), as the k increases, the model is more likely to accept longer sequences in the verification stage due to the more diverse set of candidate hypotheses, thereby significantly enhancing the acceptance rate. However, the total number of nodes in the token tree increases exponentially as the increase of k as we have discussed in Section 4. When setting k to 4, the token tree contains more than 20000 tokens which leads to a heavy computation budget. In contrast, the token graph prevents the uncontrolled swell of node count that could impede computational efficiency by merging repeating sub-trees. This optimization allows the GSD to achieve a much higher acceptance rate while free from a rapid increase in nodes with the increase of k.

Threshold for Redundant Node τ As mentioned in Section 5.2, when two different hypotheses emanating from different branches share a common token sequence of length τ, they are identified as repetition and subsequently merged as a single branch. Thus, the larger the τ, the more radical the node merging becomes. As shown in Figure 5(b), as the increase of τ, the method becomes more conservative in fusing repeated branches, retaining more nodes in the token graph. Besides, the acceptance rate is inversely correlated with the redundant threshold. This implies that more aggressive node fusion leads to a more diverse set of candidate hypotheses. At first glance, this might seem paradoxical, since one would expect that aggressive node fusion, which reduces the number of nodes in the token graph, would decrease the diversity of hypotheses by merging similar sequences. However, when the merging happens, the two nodes that are merged as one then share a common child subtree in later drafting steps. By merging, the newly generated tokens within the subtree are simultaneously added to two different branches, while these tokens might not be generated by both independent branches if not merged. Thus, the node merging effectively introduces a greater variety of hypotheses by allowing for increased sharing of information between different parts of the token graph, which might otherwise remain isolated, leading to less efficient search space coverage.

Pruning Threshold $\theta_{prob}, \theta_{sib}$ The probability pruning technique prunes tokens of low logit probability and the sibling pruning technique involves pruning sibling nodes that had passed the probability-based pruning based on the maximum logit probability. As illustrated in the figure, both pruning strategies significantly reduce the number of generated tokens. However, these two pruning strategies have opposite effects on the acceptance rate. When the threshold is raised, probability pruning leads to an increase in the acceptance rate, while sibling pruning has a diminishing effect. This indicates that while probability pruning can help in focusing the speculative decoding process on more likely hypotheses, sibling pruning might lead to the removal of potential candidate hypotheses that could have been valid. The implications of these findings suggest that a delicate balance must be struck between pruning enough to maintain computational efficiency and avoiding overly aggressive pruning that could eliminate valid hypotheses.

6.4 Non-deterministic Setting

Table 2 represents performance under the non-deterministic decoding setting. This non-deterministic verification process determines whether a drafted token should be accepted by comparing the generating probability of the draft model and the LLM. Implementing GSD in this setting is a little tricky because GSD uses a shared logit distribution for redundant tokens, which could slightly deviate from the actual distribution. We have addressed the potential effects of this issue in Section 5.3 through experimental analysis. Furthermore, we conduct an explicit evaluation of the text quality, confirming that the performance disruption due to node merging is inconsequential. Detailed results can be found in Appendix E.

7 Analysis

7.1 Breakdown of Computation

Table 3 presents a computational analysis comparing different speculative decoding methods. Compared to TSD, the primary improvement offered by GSD lies in the reduction of time consumed during the draft stage, which can be attributed to the fewer number of nodes in the token graph, resulting in a reduced count of tokens that need to be processed during each drafting forward pass.

Besides, we find that, in addition to drafting and verifying, there is a significant portion of computation that should not be overlooked. We find that this computation is primarily associated with the update of the kv-cache of the draft model. Thus, improving the efficiency of the kv-caching represents a potential direction for further accelerating the speculative decoding.

Methods	Draft	Verification	Others
SSD	224.9 ms	133.5 ms	45.8 ms
TSD(k=2)	257.0 ms	172.4 ms	46.9 ms
GSD(k=2)	225.9 ms	170.0 ms	45.5 ms
TSD(k=4)	323.9 ms	184.4 ms	49.8 ms
GSD(k=4)	209.0 ms	178.3 ms	50.2 ms

Table 3: Breakdown of computation of a **single** draft-verification iteration.

Graph-like Speculative Decoding

[Input] Write a story about life on Mars:

[Output] The Martian Chronicles is a 1950 science fiction short story collection by Ray Brad bury that chronicles the exploration and settlement of Mars in the near future. The stories range from the first expeditions to the Red Planet to the eventual demise of the human race. The collection is notable for its use of science fiction to explore social and political themes, as well as its vivid and imaginative depiction of life on Mars. The stories are also notable for their use of allegory and symbolism, as well as their exploration of the human condition. The Martian Chronicles is considered one of the most influential works of science fiction ever written and has been adapted into a number of different media, including a television series, a radio drama, and a comic book. The Martian Chronicles is a collection of short stories by Ray Brad bury that chronicles the colonization of Mars by humans.

Figure 6: A visualization of the generation process of graph-structured speculative decoding. The black color represents the token generated by the verification model. Both red and blue are the accepted tokens. Red tokens are ordinarily drafted while blue tokens are from the merged nodes of the token graph.

7.2 Case Study

Figure 6 presents an illustrative example of GSD. This case demonstrates how the token graph assists in maintaining various hypotheses while simultaneously decreasing the total number of drafted tokens. Notably, approximately 30% of the accepted drafted tokens are derived from the subtrees associated with merged nodes, illustrating the efficiency gains achieved through GSD.

8 Conclusion

In this paper, we introduce graph-structured speculative decoding (GSD), a novel decoding strategy that utilizes a token graph to concurrently record a multitude of sequence hypotheses within a single draft stage. We propose a redundant node merging technique and two pruning strategies to constrain the size of the token graph without unduly compromising the diversity of hypotheses. Our extensive experiments demonstrate that GSD significantly increases the acceptance rate of drafted tokens while not introducing much computation, achieving a noticeable acceleration in speed compared to previous speculative decoding methods.

Limitations

We discuss the limitations of our work as follows: While our investigation has highlighted an interesting phenomenon of hypotheses generated from the same context contexts, we have not thoroughly examined the underlying mechanism that gives rise to this phenomenon. A deeper exploration into why these hypotheses exhibit such close semantic ties could unveil further insights that may benefit future research and applications.

References

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

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. *CoRR*, abs/2302.01318.

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. *CoRR*, abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711.

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6524–6538, Online. Association for Computational Linguistics.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023a. LLM-QAT: data-free quantization aware training for large language models. *CoRR*, abs/2305.17888.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023b. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

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

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

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.

OpenAI. 2022. Openai chatgpt.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2021. Alp-kd: Attention-based layer projection for knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13657–13665.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Benjamin Spector and Christopher Ré. 2023. Accelerating LLM inference with staged speculative decoding. *CoRR*, abs/2308.04623.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10107–10116.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4821–4836, Dublin, Ireland. Association for Computational Linguistics.

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, 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, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

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. *CoRR*, abs/2309.08168.

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. *CoRR*, abs/2310.08461.

A Additional Implementation Details

We establish both the maximum input sequence length and output sequence length at 512. Any input sequences exceeding 512 tokens are truncated. We set the maximum drafting step at 10 and adopt a draft-exiting mechanism to prematurely exit the drafting stage when the token probability drops below θ_{prob}. For the top-p sampling decoding, we set the top-p to 0.7 and temperature to 0.7. For graph decoding and tree decoding, we set the maximum out-degree k as 4. For the pruning configurations, we default to $\theta_{prob} = 0.2$ and $\theta_{sib} = 0.3$. We set $\tau = 2$.

B Comparison with Other Inference Acceleration Methods

Methods	GSM8K	XSUM
Medusa	2.01×	1.62×
GSD	1.96×	1.73×

Table 4: Speedup ratios on LLaMA-2-70b.

Except for speculative decoding, there have been other methods for accelerating the decoding of LLM. Among these studies, Medusa (Cai et al., 2023) is a simple yet effective method. We compare with Medusa on GSM8K and XSUM (Table 4). Besides, we want to mention that Medusa is dedicated to the same deterministic setting and employs a similar tree structure to manage the generated tokens, so it is possible to incorporate Medusa with our proposed graph structure to further optimize the token management. Hopefully, this would bring further acceleration.

C Impact of Node Merging on Logits Distribution

maximum out-degree	$\tau = 0$	$\tau = 1$	$\tau = 2$	$\tau = 3$	$\tau = 4$
k=3	1.19e-4	2.70e-6	5.77e-7	3.34e-7	5.13e-7
k=5	1.78e-4	4.70e-6	4-21e-7	7.62e-7	8.49e-7
k=∞	1.30e-4	3.11e-6	1.03e-6	9.27e-7	7.64e-7

Table 5: Averaged KL-divergence between the probability distributions across the vocabulary with or without node merging. Results are averaged over 1000 examples. We test on a series of k (maximum out-degree) and τ (the threshold for redundant node), showing that in most cases, merging redundant nodes brings minimal affection to the generation probability of subsequent tokens.

D Additional Results on Deterministic Setting

We present the evaluation results on BLOOM-7b1, OPT-13b, and LLaMA-7b in Table 6,7, and 8.

Methods	Alpaca	WMT-14 en-de	gsm8k
SSD	0.628 (1.12×)	0.705 (1.30×)	0.653 (1.18×)
TSD	0.783 (0.44×)	0.798 (0.59×)	0.741 (0.32×)
GSD	0.819 (1.48×)	0.812 (1.52×)	0.755 (1.26×)

Table 6: BLOOM-7b1 performance under $k = 4$, $\tau = 1$, $\theta_{prob} = 0.4$, $\theta_{sib} = 0.1$. BLOOM-560m serves as the draft model.

Methods	Alpaca	WMT-14 en-de	gsm8k
SSD	0.563 (1.12×)	0.621 (1.16×)	0.602 (1.08×)
TSD	0.672 (0.37×)	0.705 (0.38×)	0.770 (0.62×)
GSD	0.691 (1.15×)	0.733 (1.28×)	0.793 (1.22×)

Table 7: OPT-13b performance under $k = 4$, $\tau = 1$, $\theta_{prob} = 0.4$, $\theta_{sib} = 0.1$. OPT-350m serves as the draft model.

Methods	Alpaca	WMT-14 en-de	gsm8k
SSD	0.729 (1.22×)	0.783 (1.29×)	0.601 (1.04×)
TSD	0.846 (0.65×)	0.851 (0.56×)	0.775 (0.60×)
GSD	0.860 (1.31×)	0.863 (1.36×)	0.793 (1.16×)

Table 8: LLaMA-2-7b performance under $k = 4$, $\tau = 1$, $\theta_{prob} = 0.4$, $\theta_{sib} = 0.1$. LLaMA-160m serves as the draft model.

E Additional Results on Non-deterministic Setting

Table 9 shows results on LLaMA-2-7b under the non-deterministic setting. In this scenario, the text produced by the model is not necessarily identical to that which would be generated via a standard decoding process. Consequently, to ensure that GSD does not significantly impair output quality, we assess the quality of the generated text. The results are shown in Table 10.

F Further Analysis on GSD

We present some extra explorations in this section. GSD introduces a novel directed acyclic graph structure to manage the drafted tokens. Every branch starting from the root node forms a unique hypothesis. We analyze the positional structure of the accepted/rejected nodes within the graph.

Methods	Alpaca	WMT-14 en-de	gsm8k
SSD	0.695 (1.16×)	0.737 (1.21×)	0.540 (0.96×)
GSD	0.793 (1.34×)	0.848 (1.31×)	0.836 (1.18×)

Table 9: LLaMA-2-7b performance under $k = 4$, $\tau = 1$, $\theta_{prob} = 0.4$, $\theta_{sib} = 0.1$. The hyperparameter settings might not be optimal.

	Rouge-1	Rouge-2	Rouge-l
vanilla decoding	0.25	0.09	0.19
SSD	0.24	0.09	0.19
GSD ($\tau = 1$)	0.23	0.09	0.18
GSD ($\tau = 2$)	0.23	0.09	0.18

Table 10: Rouge-1/2/l scores on LLaMA-2-7b under non-deterministic setting.

Figure 7 shows the benefit of considering multiple hypotheses in enhancing the acceptance rate, with approximately half of the accepted tokens originating from Child-k nodes (where $k > 1$). These tokens are typically not taken into account in SSD. Comparing TSD and GSD, we can see that GSD slightly increases the acceptance rate for tokens positioned as Child-1. Figure 8 shows how varying the θ_{sib} threshold impacts the acceptance rate for tokens at each position. A higher θ_{sib} corresponds to more stringent pruning, resulting in fewer sibling nodes being retained. We can see a clear negative correlation between the increase of θ_{sib} and the acceptance rate for tokens at latter positions.

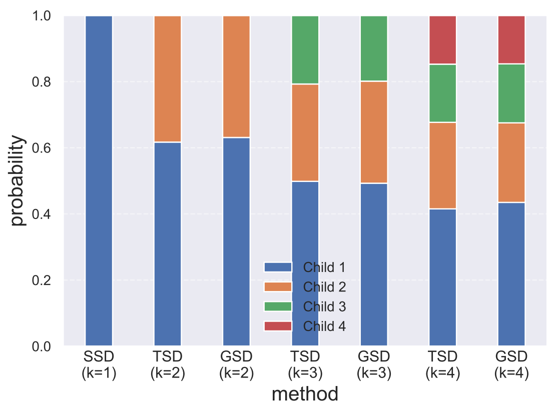

Figure 7: Percentage of i-th child being accepted. Results are averaged across all nodes within the token graph. We compare various speculative decoding configurations on LLaMA-7b. The child nodes within the token graph are ranked according to their probability, such that Child-1 corresponds to the token with the highest probability, while Child-k represents the token with the k-th highest logit probability.

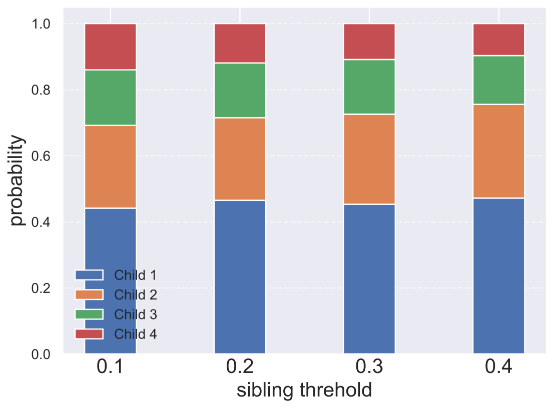

Figure 8: Percentage of i-th child being accepted. Results are obtained from LLaMA-7b with $k = 4$, $\tau = 1$, $\theta_{orob} = 0.4$

.