SLiM: Speculative Decoding with Hypothesis Reduction

Chi-Heng Lin **Shikhar Tuli** **James Seale Smith**

Yen-Chang Hsu **Yilin Shen** **Hongxia Jin**
Samsung Research America
{chiheng.lin,shikhar.tuli,james.smith,yenchang.hsu,
yilin.shen,hongxia.jin}@samsung.com

Abstract

Speculative decoding has emerged as a prominent alternative to autoregressive decoding for expediting inference in large language models (LLMs). However, prevailing assumptions often focus solely on latency reduction, neglecting the computational expenses. In this paper, we present **S**peculate **L**ess, val**i**date **M**ore (SLiM), a speculative decoding enhancement to reduce the speculation set while validating more effective tokens. SLiM is designed to mitigate LLMs' computation costs associated with the token verification by introducing hypothesis reduction based on a fast posterior estimation. It consistently surpasses counterparts lacking cost reduction across a spectrum from CPU to GPU. Our evaluation with diverse conversational datasets shows that SLiM can achieve a substantial 70% reduction in FLOPs while generating more effective predictions on top of prior arts.

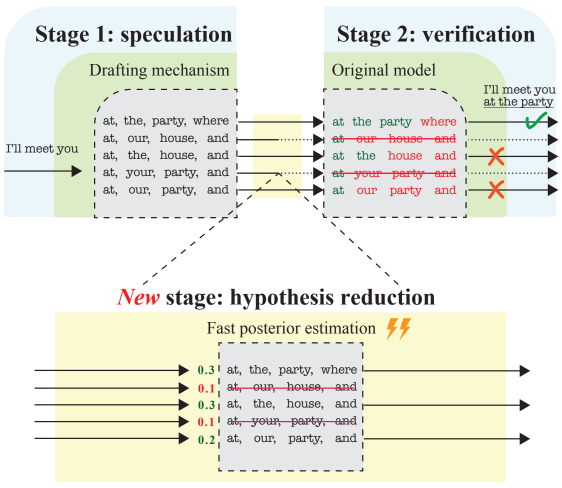

Figure 1: Prior works parallelize token generation by (1) generating multiple hypotheses with a lightweight drafting mechanism and (2) verifying the hypotheses with powerful hardware in parallel. **Our work** introduces a **hypothesis reduction** stage to **drastically reduce the computation**, making our approach friendly for resource-constrained devices.

1 Introduction

Recent advancements in large language models (LLMs), such as LLaMA (Touvron et al., 2023a,b), GPT-4 (OpenAI, 2023b), and Vicuna (Chiang et al., 2023a), have showcased their tremendous potential as proficient artificial intelligence (AI) assistants (Geng and Liu, 2023; Biderman et al., 2023) across diverse domains. Despite their widespread adoption, these models are severely limited by inference speed due to their serial decoding mechanisms.

To address this concern, several methods have been proposed to expedite token generation. In particular, *speculative decoding* (Leviathan et al., 2023; Xia et al., 2022; Miao et al., 2023; Liu et al., 2023; Spector and Re, 2023; Yang et al., 2023) has emerged as a prominent strategy by leveraging the *speculate and verify* mechanism, resulting in *substantial* inference time reduction. It is a dual-stage procedure wherein (1) lightweight models generate hypothesized token sequences speculatively, and (2) the original LLMs verify their acceptance. The key principle of speculative decoding is that it reframes the inherent *sequential* decoding of tokens as a *parallel* operation, leveraging hardware's parallelization power to reduce latency for time-sensitive applications.

Two criteria need to be met for speculative decoding to gain speed: (1) the lightweight model must draft the predictions much faster than the original LLM, and (2) the device must have sufficient computation throughput for the parallel verification of multiple hypotheses. While the first criterion (light drafting mechanism) has garnered attention and inspired several works, the second criterion (speedy verification) has been generally overlooked in the community. This becomes a substantial concern since recent methods tend to generate a relatively larger number of hypotheses (Cai et al., 2023). In many practical scenarios, the inference has very limited computational budgets, and therefore a high-complexity verification step

Table 1: Computational budget per iteration versus its inference time with 8-bit quantized Vicuna-7B on NVIDIA GeForce RTX 4090.

# Hypothesis	1	4	5	8	26	65	95
GFLOPs/iter	19.6	40.0	59.0	117.6	254.8	431.2	607.6
Time/iter (ms)	1.00	1.74	1.87	1.94	2.65	2.74	2.83

can offset the speed-up benefits gained from speculation. Table 1 shows an actual example, where the forward latency increases with FLOPs along with a larger number of hypotheses verification.

To mitigate the above caveat, we introduce the *three-stage* speculative decoding process which inserts a novel *hypothesis reduction* stage between the two existing stages. Figure 1 illustrates the process, in which our *hypothesis reduction* performs a fast posterior estimation to eliminate unlikely candidates, yielding high computation savings in the verification stage. In other words, we propose to have a lightweight verification strategy before performing the expensive one. Our lightweight verification computes the corrected posterior estimation based on a simple bigram correlation function, providing a more confident assessment to prune the hypotheses.

Figure 2 provides a preview of how our strategy significantly reduces the number of floating-point operations (FLOPs) per speculation step needed to achieve the same speed. In summary, we make the following contributions:

- Motivated by the need to reduce the number of costly verifications for real-time applications, we expand the speculative decoding paradigm with a third *hypothesis reduction* stage to achieve substantial computation savings in the verification stage.
- We contribute SLiM as a method for hypothesis reduction in speculative decoding and show that we can save 70% computations while achieving a competitive speed-up performance (1.8-$2.3\times$) on various conversational benchmarks and different model sizes, paving the way for fast LLM inference for on-device applications.
- We conduct empirical studies of SLiM on diverse devices from CPUs to powerful GPUs. SLiM universally outperforms both autoregressive decoding and state-of-the-art batch-speculative decoding in terms of real-time latency and token generations per forward pass.

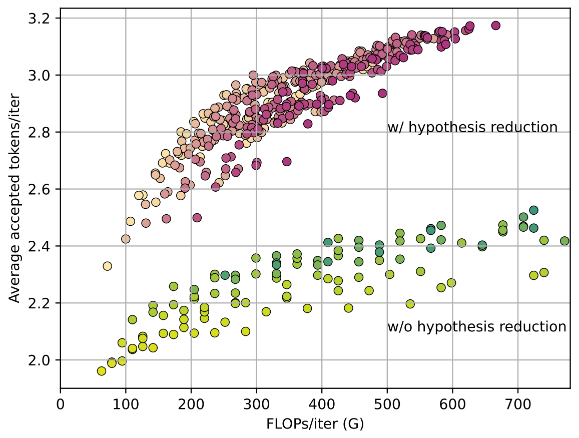

Figure 2: Speedup-computations trade-offs for LLM inference on Vicuna-80 generic prompts (Chiang et al., 2023a). The green and red points illustrate the distribution of speedup-computation trade-offs within speculative decoding, both with and without the hypothesis reduction proposed in this study. Our proposed hypothesis reduction technique increases the token acceptance rate with reduced computations.

2 Related Works

Given the widespread use of Large Language Models (LLMs) across diverse applications, the acceleration of their inference speed has garnered significant attention (Kim et al., 2023a). Extensive efforts have been dedicated to replacing the autoregressive decoding method with parallel decoding (or non-autoregressive decoding) (Gu et al., 2017; Wang et al., 2019; Li et al., 2019; Wei et al., 2019; Shao et al., 2020; Ghazvininejad et al., 2019; Guo et al., 2020; Kasai et al., 2020). While these works focus on machine translation, Welleck et al. (2019); Gu et al. (2019); Stern et al. (2019); Schuster et al. (2022) consider sentence generation task. However, these approaches often necessitate intricate model training, which limits their practical applicability in the context of large models.

Recently, Speculative Decoding (Chen et al., 2023; Leviathan et al., 2023) has stood out as a prominent approach to accelerate LLMs' inferences. This method adopts a "speculate and verify" strategy, utilizing a draft model—a smaller model with faster inference capabilities—to propose tokens for verification by the original model. This approach enables the generation of multiple tokens simultaneously, enhancing overall inference speed.

While earlier works predominantly focus on single-sequence speculation (Stern et al., 2018; Xia et al., 2022; Chen et al., 2023; Leviathan et al., 2023; Gante, 2023; Liu et al., 2023), recent advancements leverage parallel computing for batch-

Table 2: Speculative decoding methods categorized by important features.

Method	Draft and base models combined	Batch speculation	Hypothesis reduction
Chen et al. (2023)	✗	✗	✗
Xia et al. (2022)	✗	✓	✗
Stern et al. (2018)	✓	✗	✗
Cai et al. (2023)	✓	✓	✗
Ours	✓	✓	✓

sequence speculations to further accelerate inferences (Cai et al., 2023; Miao et al., 2023; Spector and Re, 2023; Yang et al., 2023). Notably, (Fu et al., 2023) introduces a novel lossless parallel decoding method based on Jacobi iteration. However, these approaches often come at the cost of increased computation due to the enlargement of the hypothesis set.

In the context of lossy acceleration methods, where the generated output deviates from the original model, BiLD (Kim et al., 2023b) employs a large model to enhance and refine the generation of a smaller model. On the other hand, SoT (Ning et al., 2023) takes a distinct approach by initially creating a skeleton of points and subsequently completing each point in parallel. SLiM differs from them by preserving the original outputs.

In contrast to existing research, SLiM incorporates the batch-speculative strategy with sequence posterior estimation to reduce the hypothesis set without compromising the acceptance rate. The estimation has been guided with a correlation function learned from online corpora. While (Yang et al., 2023) also leverages online corpora for speculation, it necessitates an exact match of context between the corpus and model inputs, limiting its applicability to specific types of problems like retrieval-augmented generation. In contrast, SLiM is application agnostic. When selecting the draft model, SLiM follows the strategy introduced by Stern et al. (2018) and Cai et al. (2023) to mitigate additional complexity. This method entails integrating additional prediction heads atop the base model, allowing it to predict extra tokens. Notably, while these approaches share a common foundation, neither incorporates any hypothesis reduction technique. SLiM stands out as the first method specifically designed to reduce the complexity of both speculative and verification stages. Table 2 provides an overview of SLiM's position in the speculative decoding landscape alongside representative examples.

3 Method

3.1 Background

Speculative decoding endeavors to predict the next m tokens $\mathbf{x}_{1:m}$ simultaneously, given an input prompt $\mathbf{h}$. It comprises two sequential stages in each iteration: (i) speculate and (ii) verify.

In the speculate stage, the method speculates sequences of the next m tokens. This involves relying on a draft model g to approximate the distribution of the next m tokens, expressed as $g(\mathbf{h}) \approx p(\mathbf{x}_{1:m}|\mathbf{h})$ and generate a set of hypothesized sequences $\mathcal{H}$. Moving on to the verify stage, the original model f comes into play to validate each guess. To ensure consistency with autoregressive decoding, the method identifies the longest sequence $\mathbf{x}_{1:l}$ satisfying the condition:

$$\mathbf{x}_j = \arg\max f([\mathbf{h}, \mathbf{x}_{1:j-1}]),\ j = 1, \ldots, l, \quad (1)$$

among all hypothesized sequences $\{\mathbf{x}_j\}_j \in \mathcal{H}$.

This paradigm involves a trade-off between the two stages. An accurate draft model can generate more accurate guesses, resulting in a smaller $\mathcal{H}$ and fewer sequences needing verification by the original model during the verification stage. However, designing a powerful draft model is nontrivial, and a complex model might incur significant latency during the forward call of $g(\mathbf{h})$, offsetting the benefits of multi-token acceleration in real time. Conversely, a simple draft model is easier to obtain but may necessitate a larger set $\mathcal{H}$ for verification, leading to higher hardware requirements of parallel computing and increased latency.

3.2 SLiM: Speculate Less and Validate More

SLiM proposes a solution to the dilemma that alleviates computation burdens in both the speculate and verify stages by adopting a simple draft model and a hypothesis reduction technique. In the speculate stage, rather than directly approximating the joint distribution of $\mathbf{x}_{1:m}$, SLiM trains $m-1$ additional prediction heads atop the same backbone f to predict the marginal distributions $p(\mathbf{x}_i|\mathbf{h})$, where $i = 2, \ldots, m$. This results in a draft model comprising $m-1$ models approximating the marginals, $g_i(\mathbf{h}) \approx p(\mathbf{x}_i|\mathbf{h})$, with each g_i having a similar complexity to the original prediction head of f.

The hypothesis set $\mathcal{H}$ is constructed through the Cartesian product of the top-k selection for each g_i

alongside the top-1 prediction of f:

$$\mathcal{H} = \text{Top}_1(f(\mathbf{h})) \times \prod_{i=2}^{m} \text{Top}_k(g_i(\mathbf{h})), \quad (2)$$

where $\text{Top}_k(p)$ denotes the set of elements with the top k probabilities in the distribution p. Notably, the set's size grows exponentially as k^{m-1}.

SLiM introduces a hypothesis reduction technique by estimating the posterior probability $p(\mathbf{x}_{1:m}|\mathbf{h})$ for each sequence and retaining those with the top k probabilities. Given that decoding with maximum posterior is Bayes' optimal, this approach ensures that accuracy is not compromised. Furthermore, the hypothesis size is reduced from an exponential to a linear function of the top-k parameter. This reduction empowers SLiM to explore a larger hypothesis set without the necessity of verifying every hypothesis with the LLM. Consequently, SLiM effectively engages in less speculation while validating more tokens.

The posterior is estimated by the formula:

$$\hat{p}(\mathbf{x}_{1:m}|\mathbf{h}) = \prod_{i=1}^{m} p(\mathbf{x}_i|\mathbf{h}) \times \prod_{j=1}^{m-1} r(\mathbf{x}_j, \mathbf{x}_{j+1}), \quad (3)$$

where $r : V \times V \mapsto \mathbb{R}$ is some fixed correlation function of two adjacent tokens, and $p(\mathbf{x}_i|\mathbf{h})$ can be approximated by the prediction heads $g_i(\mathbf{h})$. The first term in equation 3 represents joint estimation under the assumption of token independence, while the second term enhances estimation with bigram.

For a visual representation of our posterior estimation, we employ Fig. 3. In this illustration, the hypothesis $\mathcal{H}$ is constructed by combining the top-2 and top-3 tokens predicted by the 2nd and 3rd prediction heads. These combinations are represented as nodes in a tree. Each edge is assigned a weight equivalent to the correlation r calculated by the endpoint nodes, and each node is associated with the probability assessed by the prediction heads. The posterior probability of a node is determined by the product of all node probabilities and edge weights along the path leading to it. The complete SLiM method is summarized in Algorithm 1.

Theoretical Interpretation: SLiM's posterior estimation is a problem of joint distribution estimation based on marginals (Frogner and Poggio, 2019). An effective strategy involves leveraging a multimarginal variant of the optimal transport problem (Peyré et al., 2019; Séjourné et al., 2019).

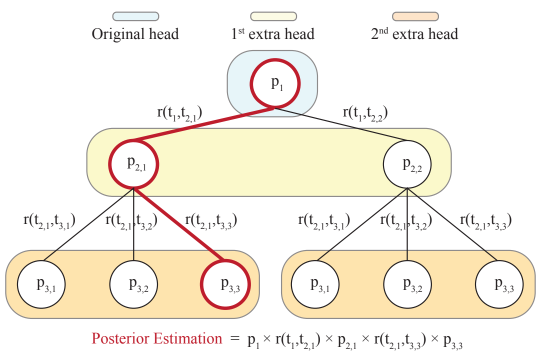

Figure 3: Illustration of SLiM's posterior estimation.

Algorithm 1 SLiM

```
Require: input prompt $\mathbf{h}$, original model $f$, marginal draft models $g_i$, $i = 2, \ldots, m$, correlation function $r$, top-k parameter $K$.
1: while stop criteria not met do
2:     Form the hypothesis set $\mathcal{H}$ in equation 2.
3:     $\forall \mathbf{x}_{1:m} \in \mathcal{H}$, obtain posterior by equation 3.
4:     Form the pruned set $\widetilde{\mathcal{H}}$ by choosing elements with largest K posteriors in $\mathcal{H}$.
5:     Choose the sequence $\mathbf{x}_{1:l}$ in $\widetilde{\mathcal{H}}$ that has the longest length satisfying equation 1.
6:     Concatenate the prompt $\mathbf{h} \leftarrow [\mathbf{h}, \mathbf{x}_{1:l}]$.
7: end while
```

Multimarginal optimal transport is a technique designed to identify the joint distribution $p(\mathbf{x}_{1:m})$ that minimizes the cost of assembling a sequence. Formally, given marginal distributions $p_i(\mathbf{x}_i)$ and a cost $c(\mathbf{x}_{1:m})$ associated with forming the sequence $\mathbf{x}_{1:m}$, the joint distribution is determined as the solution to the constrained optimization problem:

$$\min_p \int p(\mathbf{x}_{1:m})c(\mathbf{x}_{1:m}) + \text{KL}\left(p || \bigotimes_{i=1}^{m} p_i\right), \quad (4)$$

with constraints that p's marginals equal to p_i. In SLiM, we relax these constraints for faster computation. The following theorem establishes the equivalence of the solution to SLiM's posterior estimation in equation 3. The proof is available in Appendix A.

Theorem 1. *Suppose the optimal transport cost is $c(\mathbf{x}_{1:m}) = -\sum_{j=1}^{m} \log r(\mathbf{x}_j, \mathbf{x}_{j+1})$, the relaxed solution to the optimization equation 4 is the posterior estimation in equation 3, subject to a normalization constant.*

3.3 Implementation

In our study, we focus on transformer-based language models and adopt prediction-head-based draft models following the approach used in Stern et al. (2018); Xia et al. (2022); Cai et al. (2023).

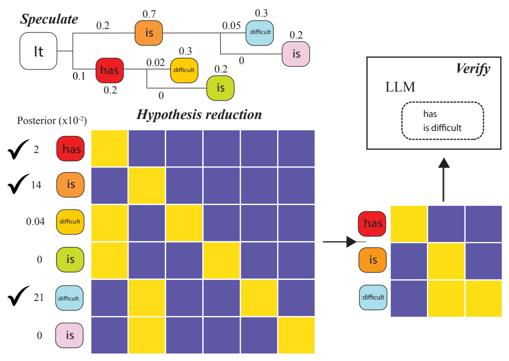

Figure 4: Tree attention with hypothesis reduction. Left and right visualize the attention mask before and after reduction.

Specifically, each extra head consists of two linear layers with a SiLU activation in between them. The first linear layer has an input and output size equivalent to the token embedding size, while the second linear layer has an output size equivalent to the vocabulary size. A skip connection was also introduced before the first linear layer and after the SiLU activation.

To construct the correlation function r, we estimate the prior distribution of two adjacent tokens by $p(\mathbf{x}, \mathbf{y}) := \frac{n(\mathbf{x},\mathbf{y})}{n}$ and the correlation function is defined as follows:

$$r(\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) \sum_{\mathbf{x}} p(\mathbf{x}, \mathbf{y})}. \quad (5)$$

The correlation can be presented as a sparse matrix with a shape of $\mathbb{R}^{V \times V}$ for storage efficiency.

For efficient verification of batch-speculated sequences, we leverage the tree attention mechanism used in the work of Cai et al. (2023). The strategy flattens the tree representation (see Figure 3) for the whole set of hypotheses and encodes its structure through the attention mask. This method effectively avoids duplicated computation of the common prefix among similar hypotheses. We adopt this strategy and convert our pruned tree with the same tree attention mechanism. Figure 4 provides a visual example of our verification reduction combined with tree attention, where the left and right heatmaps illustrate the mask before and after the hypothesis reduction. Note that the attention masks in our work are dynamically computed since the tree has a dynamic shape. This is significantly different from the implementation of Cai et al. (2023), which uses a static mask designed heuristically.

4 Results

4.1 Experimental Setup

Models. We employ the Medusa model (Cai et al., 2023), built upon the Vicuna-7B and Vicuna-13B (Chiang et al., 2023a) as base models. Additionally, the model is trained on the public ShareGPT dataset to incorporate four additional prediction heads, enabling speculation on a maximum of four additional tokens. To derive the correlation function r for SLiM's posterior estimation, we record the frequency of adjacent tokens in the LMSYS-Chat-1M dataset (Zheng et al., 2023). The resulting r is stored in the sparse format, amounting to 82MB in size. We implement our framework on top of PyTorch (Paszke et al., 2019) and the HuggingFace Transformers library (Wolf et al., 2019).

Datasets. We evaluate SLiM's generation capabilities using prompts from six conversational datasets: Vicuna-80 (Chiang et al., 2023b) includes nine different categories of prompts: (CF), coding (CD), knowledge (KL), generic (GN), fermi (FM), roleplay (RP), writing (WT), common sense (CS) and math (MA), and five datasets, each having 360 to 1000 prompts: Chatbot Instruction Prompts (CIP) (Palla, 2023), ChatGPT Prompts (CP) (OpenAI, 2023a), WebQA (Berant et al., 2013), Alpaca (Taori et al., 2023; Peng et al., 2023), and PIQA (Bisk et al., 2020).

Environments. We test the performance on three devices, spanning a spectrum of computation power: Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz with 20 cores, single NVIDIA RTX 4090 24GB GPU, and single NVIDIA A100 80GB GPU. Except for the A100, all models undergo testing with 8-bit quantization (Dettmers et al., 2022) to ensure compatibility with RAM constraints.

Comparing methods. For a fair comparison, we evaluate SLiM's performance against other speculative methods that use extra prediction heads to generate hypotheses. All methods utilize the same public models, Vicuna-7B and Vicuna-13B, with the same set of prediction heads trained by Cai et al. (2023). We examined two extremes in speculative methods: block parallel decoding (BPD) (Stern et al., 2018) and Medusa (Cai et al., 2023). BPD opts for a single-sequence speculation per iteration, while Medusa adopts batch speculation, concurrently verifying multiple sequences through an optimized tree attention mechanism. SLiM positions

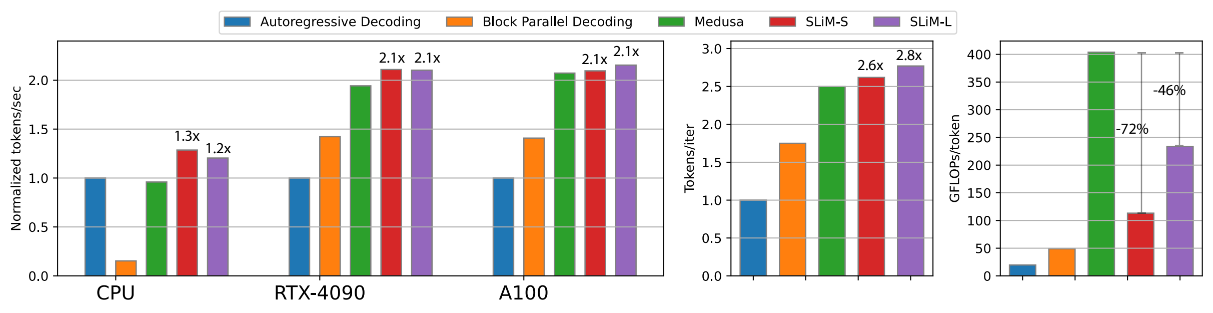

Figure 5: Inference acceleration and FLOPs consumptions for prediction-head-based speculative methods on various devices.

Table 3: Inference results of prediction-head based speculative methods on single RTX 4090 GPU across diverse conversational datasets. Left: real-time accelerations (in tokens per second), with speed-up multipliers relative to the autoregressive decoding. Right: GFLOPs consumption, with percentage relative to Medusa.

	Inference Speed (tokens/s)						GFLOPs per token					
Dataset	CIP	CP	WebQA	Alpaca	PIQA	Avg.	CIP	CP	WebQA	Alpaca	PIQA	Avg.
Autoregressive Decoding	5.56	5.69	5.66	5.76	5.72	5.68 (1.00x)	27.74	25.89	23.23	24.09	18.45	23.58
BPD (Stern et al., 2018)	8.33	7.97	7.47	8.35	8.39	8.12 (1.43x)	56.26	58.48	61.63	55.3	51.49	56.36
Medusa (Cai et al., 2023)	10.51	9.47	8.85	10.05	10.37	9.91 (1.74x)	401.63	417.53	456.55	454.58	426.89	433.38
SLiM-S	**11.13**	**10.09**	**9.19**	**10.46**	**10.52**	**10.31** (**1.82x**)	124.21	134.88	145.10	124.54	126.42	130.46 (30%)

itself between these extremes with reduced batch speculation. We evaluate three configurations of SLiM, denoted as SLiM-SS, SLiM-S and SLiM-L, which have 1, 10 and 20 average hypothesis sequences, respectively. They reduce computation by roughly 70% and 40% as compared to Medusa, respectively. The detailed configurations can be found in Appendix C.

Metrics. We investigate three important metrics: (i) Real-time acceleration: quantified in tokens/s, this metric represents the average number of tokens generated per second. (ii) Device-agnostic acceleration: measured in tokens/iter, this metric reflects the average number of tokens verified or generated with each forward call of the base model. (iii) Computation consumption per effective token generation: denoted as GFLOPs/token, this metric signifies the average number of floating-point operations required to generate a valid token.

4.2 Generation speed-up experiments

The key advantage of SLiM is in mitigating computation burdens without compromising the accuracy of multi-token predictions, making it versatile across devices with diverse computation capabilities. To substantiate this claim, our experiments encompassed three distinct devices in Fig. 5. It delineates real-time acceleration, token generation per iteration, and FLOPs consumption, utilizing Vicuna-7B as the base model on the Vicuna-80 dataset. Notably, on CPUs, SLiM stands out as the sole method achieving real-time speed-up, despite all approaches showcasing device-agnostic acceleration. The gap between real-time and device-agnostic acceleration on CPUs is attributed to the limited parallel computation capability, resulting in significant latency when verifying a large number of batch speculations—offsetting the advantages of multi-token prediction. While BPD incurs lower computation costs, its verification acceptance rate is modest, and Medusa achieves a high acceptance rate at the expense of excessive computation costs.

In contrast, as depicted in the middle and right figures, SLiM-S and SLiM-L generate a comparable number of tokens while utilizing only 28% and 58% of the computations required by Medusa. These findings underscore the significance of hypothesis reduction, especially in applications with constrained computation power. Achieving an optimal balance between the number of speculations and the verification acceptance rate becomes crucial in such scenarios.

On GPUs, Medusa and SLiM demonstrate comparable performance, outperforming BPD due to their batch speculations and efficient parallel verification. Table 3 zooms in on GPU evaluation for five additional datasets, testing on a single GTX 4090. Notably, SLiM surpasses Medusa with only 30% of the computations. As SLiM introduces a hypothesis reduction scheme distinct from Medusa, these results underscore the consistent enhancement the scheme provides across diverse environments in computation and inference speed.

We conducted an analysis of various model sizes

Table 4: Inferences on Vicuna-80 with varying model sizes. Left: real-time accelerations, with speed-up multipliers relative to autoregressive decoding. Right: FLOPs consumption, with percentage relative to the batch-speculative method Medusa.

	Inference speed (Tokens/s)										Tokens/iter	GFLOPs/token
Model	CF	CD	KL	GN	FM	RP	WT	CS	MA	AVG		
Vicuna-7B	5.87	6.1	5.96	5.97	6.05	5.99	6.00	5.93	6.15	5.99	1.00	19.61
BPD-7B (Stern et al., 2018)	6.58	9.49	8.02	9.74	8.53	7.79	9.09	8.66	10.37	8.52 (1.42×)	1.75	48.85
Medusa-7B (Cai et al., 2023)	10.6	14.75	11.01	12.83	10.91	10.07	11.79	11.21	14.47	11.63 (1.94×)	2.50	403.91
SLiM-S-7B	11.13	**16.68**	**11.68**	**13.92**	**11.95**	**10.85**	**12.91**	**12.26**	**15.64**	**12.63 (2.11×)**	2.58	113.23 (28%)
SLiM-L-7B	**11.23**	16.63	11.65	13.84	11.88	10.78	12.82	12.19	15.58	12.59 (2.10×)	**2.77**	233.76 (58%)
Vicuna-13B	4.4	4.48	4.38	4.31	4.41	4.45	4.4	3.99	3.63	4.32	1.00	37.92
BPD-13B (Stern et al., 2018)	6.23	9.3	6.65	7.51	6.57	6.03	7.21	6.73	7.89	6.98 (1.62×)	1.81	90.01
Medusa-13B (Cai et al., 2023)	8.36	11.41	9.1	9.94	8.77	7.84	9.28	8.75	10.79	9.16 (2.12×)	2.59	695.03
SLiM-S-13B	8.55	11.66	9.55	10.46	9.22	8.09	9.8	8.99	9.92	9.47 (2.19×)	2.64	208.89 (30%)
SLiM-L-13B	**8.84**	**12.9**	**9.9**	**11.1**	**9.56**	**8.55**	**10.5**	**9.61**	**11.29**	**10.06 (2.33×)**	**2.88**	427.27 (61%)

Table 5: Inferences on Vicuna-80 compared with different methods. We compare with the original speculative decoding (Chen et al., 2023) implemented by Huggingface's assisted generation. We tested against speculative decoding (SD) with draft models of various sizes. The results indicate that prediction-head-based draft models (BPD, Medusa, SLiM) consistently outperform the original speculative decoding in wall-clock inference speed.

	Speculation batch	Wallclock Speedup (Tokens/s)	Theoretical Speedup(Tokens/iter)	GFLOPs/token	Parameter Overhead (#)
Vicuna-13B	1	4.32	1	37.92	0
BPD-13B (Stern et al., 2018)	1	6.98 (1.62×)	1.81	90.01	760M
Medusa-7B (Cai et al., 2023)	42	9.16 (2.12×)	2.59	695.03	760M
SD-SS-13B(Chen et al., 2023)	1	5.67 (1.31×)	2.02	114.54	68M
SD-S-13B(Chen et al., 2023)	1	3.83 (0.89×)	2.43	111.91	160M
SD-L-13B(Chen et al., 2023)	1	3.53 (0.82 ×)	3.45	115.53	1.1B
SLiM-SS-13B	1	7.61 (1.76×)	2.17	89.00	760M + 82M
SLiM-S-13B	10	9.47 (2.19×)	2.64	208.89	760M + 82M
SLiM-L-13B	26	10.06 (2.33×)	2.88	427.27	760M + 82M

and prompt categories, and the findings are summarized in Table 4. The results indicate improved speed-ups for larger models, highlighting the growing significance of speculative decoding in scenarios where there is more room to trade speed-up for computational resources. Furthermore, the analysis indicates that acceleration is particularly effective for coding and math problems. This observation suggests that responses to these types of questions may be more amenable to multi-step ahead predictions than linguistic inquiries, possibly due to their formalizable nature.

In Table 5, we conducted experiments to compare with the original speculative decoding (Chen et al., 2023) implemented by Huggingface's assisted generation. The setup is the same as that of Table 4, using Vicuna-13B as the target model. We tested against speculative decoding with draft models of various sizes (68M, 160M, 1B denoted as SD-SS, SD-S, SD-L, respectively) from Huggingface. We observe that despite the speculative decoding achieves impressive theoretical speedup, it does not necessarily translate into good real-time acceleration due to the overheads in complex draft model computations.

The results indicate that prediction-head-based draft models (BPD, Medusa, SLiM) consistently outperform the original speculative decoding in wall-clock inference speed while incurring small overheads from additional prediction heads (760M) and sparse bigram matrix (82M).

4.3 Model analysis

In this section, we delve into a detailed exploration of SLiM's acceleration capabilities, focusing on experiments conducted on the Vicuna-80 dataset.

Effectiveness of equation 3: How does the correlation change the posterior estimation? Fig. 6 visually illustrates the transformation of the posterior through correlation adjustment in SLiM. In this illustrative example, we employ a specific prompt: "...Here are some tips to get you started: 1. Prioritize tasks:". The base model predicts 'Make' as the next token, while simultaneously, the prediction head generates the distribution for the subsequent token, depicted by the red line in Fig. 6. The noticeable misalignment between the red curve and the ground-truth distribution (depicted in blue) highlights the need for adjustment.

The correlation function, denoted by the grey curve, assigns a high value to 'Make a'. This weight, when multiplied with the posterior, results in the adjusted estimation denoted by the dotted red curve. Remarkably, the distribution now aligns closely with the ground truth, and its prediction of 'a' becomes an accepted outcome. Table 6 provides a quantitative record of the accuracy for the top 1 prediction from the second extra head. The results unequivocally demonstrate that the correlation function consistently enhances accuracy across all categories.

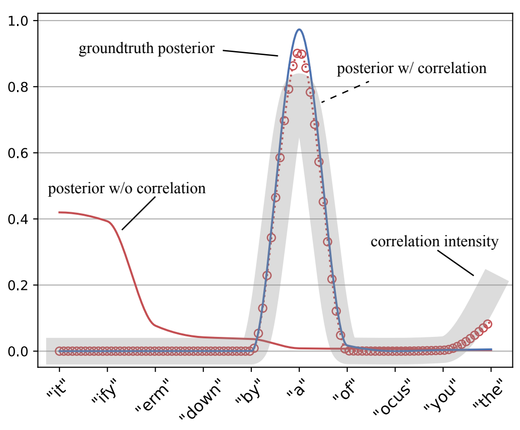

Figure 6: Posterior estimations with and without correlation adjustment.

Table 6: Extra-token prediction accuracies with and without correlation on Vicuna benchmark.

Category	CF	CD	KL	GN	FM	RP	WT	CS	MA	AVG
w/ corr.	43%	61%	49%	57%	47%	43%	52%	50%	59%	50%
w/o corr.	**52%**	**66%**	**59%**	**66%**	**54%**	**50%**	**61%**	**60%**	**66%**	**59%**

When is the correlation is helpful? While correlation is a valuable tool for posterior adjustment, as depicted in Fig. 6, it is essential to acknowledge its potential to mislead the prior estimation generated by the prediction heads. To gain a comprehensive understanding of when correlation can provide a positive impact, we conduct an analysis to assess its effectiveness.

To this end, we present SLiM's speed-ups on the Vicuna-80 dataset, utilizing correlation functions learned from various numbers of sentences from the LMSYS-Chat-1M dataset (Zheng et al., 2023), as illustrated in Fig. 7. Notably, we observe a steady increase in speed-up as the number of prompts grows, reaching saturation at 2.6× when the number exceeds 50,000. This finding underscores the positive correlation between the number of sequences and the improvement in speed-up, indicating that learning from a larger corpus is indeed beneficial. However, the improvement becomes marginal beyond a certain threshold.

To determine when correlation may be detrimental, we compare the results with hypothesis reduction that uses conditional independent predictions (i.e., $r = 1$ in equation 3), yielding a speed-up of 2.26×. Consequently, we identify 2,000 as the threshold prompt number that demarcates the regions into helpful (depicted in green) and harmful (depicted in red) in the figure. This insight suggests

Table 7: Speed-ups vs. number of extra prediction heads.

	SLiM				Medusa
# Extra heads	1	2	3	4	4
Counterfactual	1.78	2.18	**2.31**	2.29	2.29
Coding	1.88	2.55	2.94	**3.28**	3.07
Knowledge	1.80	2.24	**2.48**	2.43	2.39
Generic	1.84	2.42	2.86	**2.96**	2.76
Fermi	1.78	2.25	2.41	**2.43**	2.32
Roleplay	1.75	2.12	**2.32**	2.19	2.23
Writing	1.78	2.32	2.58	**2.64**	2.49
Common Sense	1.84	2.30	2.51	**2.57**	2.48
Math	1.89	2.47	2.73	2.92	**3.07**
Average speed-up	1.81	2.30	2.54	**2.58**	2.50
FLOPs/token	**51.52**	109.52	125.49	113.23	403.91

Table 8: Method ablation study.

Model	Batch speculation	Hypothesis reduction	With Correlation	Tokens per iter	GFLOPs per token
BPD	✗	✗	✗	1.75	48.85
Medusa	✓	✗	✗	2.50	403.91
SLiM-S	✗	✗	✓	1.97	**44.37**
SLiM-S	✓	✓	✗	2.26	124.18
SLiM-S	✓	✓	✓	**2.58**	113.23

that sufficient sentences are necessary for constructing an effective correlation function.

How many heads do we need? Moving forward, we delve into an analysis of speed-up concerning the number of prediction heads in Table 7. The results show a higher speed-up with more heads. However, the most efficient computation occurs when having only one extra head, and the efficiency diminishes as we increase the number of heads. This observation prompts considerations for selecting a proper number of heads for resource-constrained devices.

How does the posterior choose sequences with different length? Since the correlation function in equation 5 is unbounded, it can exceed 1, implying that shorter candidates are not always favored over longer ones. In general, one can also explicitly penalize shorter sequence by adding a regularization to our selection stage.

4.4 Method ablation

We conducted a comprehensive study on various SLiM variants to understand the impact of different components on speed-ups. Three key factors were considered: (i) whether to speculate multiple sequences, (ii) whether to adopt hypothesis reduction, and (iii) whether to use correlation or rely on prediction heads alone for estimating posteriors in the reduction stage. Furthermore, average results for BPD and Medusa on the entire Vicuna-80 dataset are presented in Table 8, where BPD and Medusa

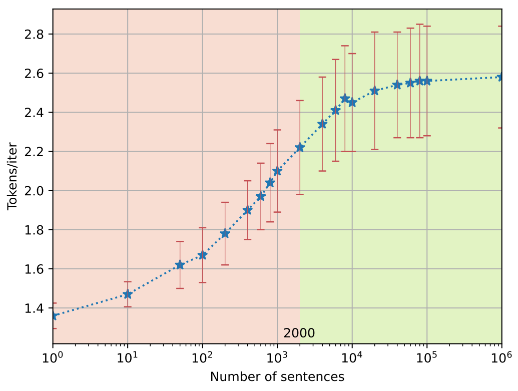

Figure 7: Speed-ups vs. the amount of corpus used for correlation training. The green region represents the cases where the resulting r is better than conditional independent predictions.

denote specific configurations of SLiM with particular choices for these factors.

The results reveal that batch speculation provides the most significant acceleration boost, with all configurations exceeding 2.2× speed-ups, whereas configurations without batch speculation exhibit speed-ups below 2.0×. However, batch speculation introduces a substantial increase in computations. The second-to-last row demonstrates that hypothesis reduction alone reduces computations to 30%, corresponding to approximately 2.2× the computation required for a single speculation, at the expense of 10% reduction in speed-up. The final row further illustrates that correlation not only maintains a similar level of computation but also enhances speed-ups. Notably, even for single speculation, the correlation alone yields a non-negligible speed-up, corroborating the evidence in Table 6.

5 Conclusions

In this work, we introduced SLiM, a speculative decoding enhancement framework designed to alleviate the computational burden associated with token verification. Our method leverages sequence posterior estimation as a lightweight verifier by incorporating bigram information. Starting with an exponentially large speculative set, we judiciously eliminate most speculations with low posteriors. Subsequently, only the sequences in the reduced hypothesis set are verified using the LLM. This approach allows SLiM to speculate fewer sequences while validating more tokens.

Empirically, our results demonstrate that SLiM's acceleration surpasses alternative methods lacking this augmentation across diverse devices. On an RTX 4090, SLiM achieves a notable 1.8-2.3× speed-up across various conversational datasets.

6 Limitations

While SLiM presents a significant improvement in terms of reduced computational requirements and enhanced speed-ups compared to conventional speculative decodings, there are still two primary challenges that warrant attention:

- Computation consumption: Despite its advantages, SLiM's computation remains higher than the autoregressive method. In scenarios where power is a critical constraint, SLiM may still underperform the baseline. Achieving optimal trade-offs between reduced FLOPs and increased speed-up compared to autoregressive methods may require an aggressive and extremely accurate hypothesis reduction strategy.

- Prediction head efficiency: Although SLiM's approach is relatively straightforward, it necessitates additional training of prediction heads. Our empirical results indicate that prediction heads are less accurate when predicting tokens for further steps and therefore may require more complex heads to be trained. Addressing this challenge involves designing efficient heads with minimal capacity while maintaining high predictive accuracy.

These challenges underscore potential areas for further research and optimization to enhance the overall effectiveness of SLiM in various application scenarios.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the International Conference on Machine Learning*, pages 2397–2430.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.

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

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

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

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

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

Charlie Frogner and Tomaso Poggio. 2019. Fast and flexible inference of joint distributions from their marginals. In *International Conference on Machine Learning*, pages 2002–2011. PMLR.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2023. Breaking the sequential dependency of llm inference using lookahead decoding.

Joao Gante. 2023. Assisted generation: a new direction toward low-latency text generation.

Xinyang Geng and Hao Liu. 2023. OpenLLaMA: An open reproduction of LLaMA.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 32.

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A Smith. 2020. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. *arXiv preprint arXiv:2006.10369*.

Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W Mahoney, et al. 2023a. Full stack optimization of transformer inference: a survey. *arXiv preprint arXiv:2302.14017*.

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

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.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. *arXiv preprint arXiv:1909.06708*.

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

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

Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-Thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337*.

OpenAI. 2023a. ChatGPT.

OpenAI. 2023b. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alessandro Palla. 2023. chatbot instruction prompts.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

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

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Thibault Séjourné, Jean Feydy, François-Xavier Vialard, Alain Trouvé, and Gabriel Peyré. 2019. Sinkhorn divergences for unbalanced optimal transport. *arXiv preprint arXiv:1910.12958*.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 198–205.

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

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, , and Tatsunori B. Hashimoto. 2023. alpaca: An instruction-following llama model.

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

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

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5377–5384.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, Jun Xie, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1906.02041*.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *International Conference on Machine Learning*, pages 6716–6726. PMLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Heming Xia, Tao Ge, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2022. Speculative decoding: Lossless speedup of autoregressive translation.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2023. Lmsys-chat-1m: A large-scale real-world llm conversation dataset.

Supplementary Material

A Proof of Theorem 1

Proof of Theorem 1. Since p is a probability distribution, it satisfies $\int p(\mathbf{x}_{1:m}) d\mathbf{x}_1 \ldots d\mathbf{x}_m = 1$. Coupling this with the objective, we obtain the Lagrangian of the objective by

$$
\begin{aligned}
L = & \int p(\mathbf{x}_{1:m})\, c(\mathbf{x}_1, \ldots, \mathbf{x}_m) d\mathbf{x}_1 \ldots d\mathbf{x}_m \\
& + \mathrm{KL}\left(p(\mathbf{x}_{1:m}) || \prod_{i=1}^{m} p_i(\mathbf{x}_i) \right) + \lambda \left(\int p(\mathbf{x}_{1:m}) d\mathbf{x}_1 \ldots d\mathbf{x}_m - 1 \right) \\
= & \int p(\mathbf{x}_{1:m})\, c(\mathbf{x}_1, \ldots, \mathbf{x}_m) d\mathbf{x}_1 \ldots d\mathbf{x}_m + \int p(\mathbf{x}_{1:m}) \log \left(\frac{p(\mathbf{x}_{1:m})}{\prod_{i=1}^{m} p_i(\mathbf{x}_i)} \right) d\mathbf{x}_1 \ldots d\mathbf{x}_m \\
& + \lambda \left(\int p(\mathbf{x}_{1:m}) d\mathbf{x}_1 \ldots d\mathbf{x}_m - 1 \right).
\end{aligned}
$$

Setting the derivative of L w.r.t. $p(\mathbf{x}_{1:m})$ to zero for every $\mathbf{x}_{1:m}$, we get

$$
\begin{aligned}
& c(\mathbf{x}_{1:m}) + \log \left(\frac{p(\mathbf{x}_{1:m})}{\prod_{i=1}^{m} p_i(\mathbf{x}_i)} \right) + 1 + \lambda = 0, \\
\Rightarrow\ & p^*(\mathbf{x}_{1:m}) \propto \prod_{i=1}^{m} p_i(\mathbf{x}_i) \times \exp\left(-c(\mathbf{x}_{1:m})\right) = \prod_{i=1}^{m} p_i(\mathbf{x}_i) \times \prod_{j=1}^{m-1} r(\mathbf{x}_j, \mathbf{x}_{j+1}).
\end{aligned}
$$

□

B Additional Experiments

We exhaustively explore different configurations of SLiM, considering various combinations of the top-k predictions in the heads. We plot the average accepted tokens with and without correlation and hypothesis reduction on the generic category prompt of Vicuna-80. The green points without hypothesis reduction exhibit the least efficiency in terms of the trade-off between the number of accepted hypotheses and efficiency. They are followed by hypothesis reduction without using correlation, with comparable pareto boundaries that require more careful tuning when hypothesis reduction is not utilized. Meanwhile, configurations with correlation in hypothesis, represented by red points, achieve optimal efficiency, outperforming speed-ups compared to Medusa.

C Detailed Configurations of SLiM

We implement SliM-SS, SLiM-S and SLiM-L by choosing top-1, top-20, and top-40 sequences based on the estimated posterior in equation 3. Since many short sequences are prefixes for longer ones, the final number of sequences to be verified is around 10 and 26 on average. Note that the formula in equation 3 does not always favor shorter sequences as the posterior is not *normalized*. As such, the correlation function can be larger than 1, making a longer sequence have a higher score.

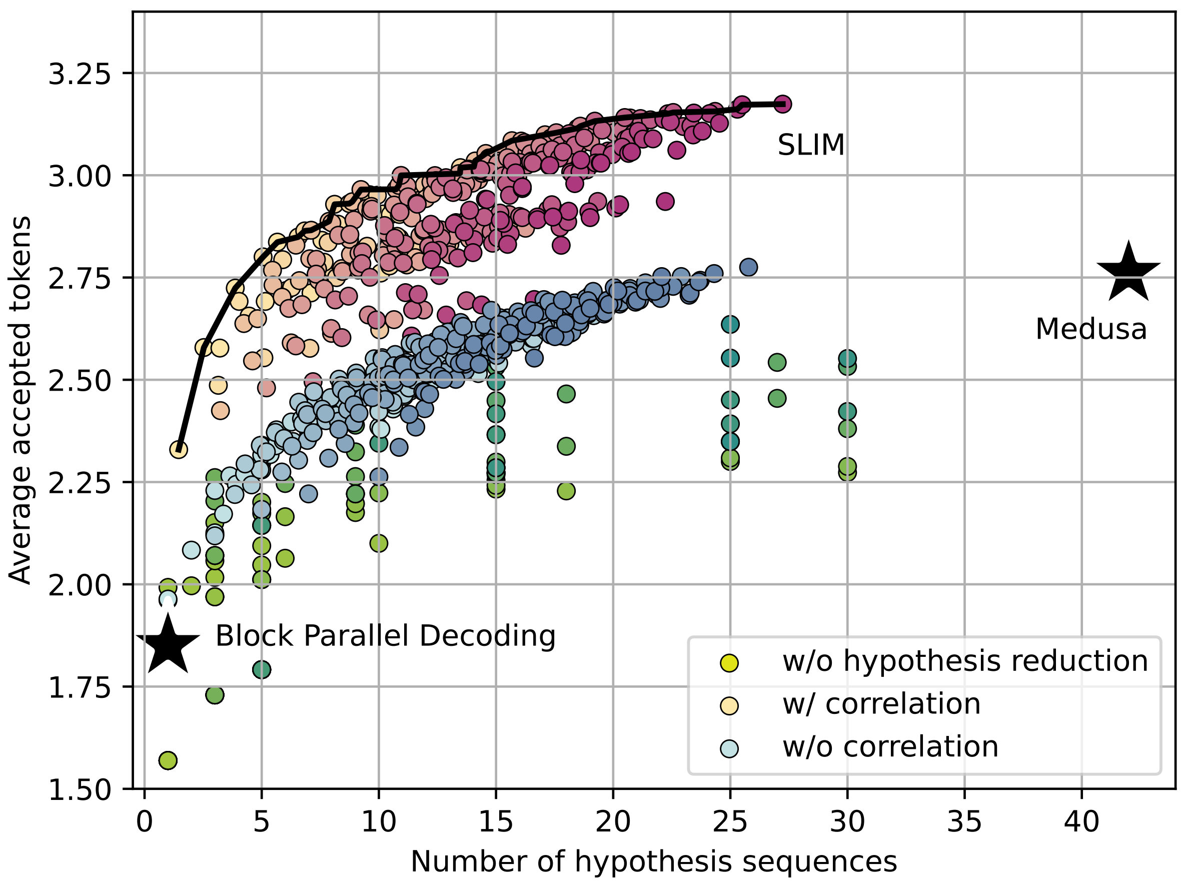

Figure 8: Average accepted tokens for various numbers of hypothesis sequences. The black line denotes the Pareto efficiency achieved by SLiM.