# Block Verification Accelerates Speculative Decoding

**Ziteng Sun** [1] **Uri Mendlovic** [1] **Yaniv Leviathan** [1]
**Asaf Aharoni**† [1] **Ahmad Beirami**† [1] **Jae Hun Ro**† [1] **Ananda Theertha Suresh**† [1]

## Abstract

Speculative decoding is an effective method for lossless acceleration of large language models during inference. It uses a smaller model to draft a block of tokens which are verified in parallel by the large model, and provides a guarantee that the output is distributed identically to a sample from the large model. In prior works, draft verification is performed token-by-token independently. Surprisingly, we show that this approach is not optimal. We propose *block verification*, a simple, easy-to-implement improved draft verification algorithm that provides additional wall-clock speedup by verifying the entire block jointly. We prove that the proposed mechanism is optimal in the expected number of tokens produced each iteration and specifically is never worse than the standard token-level verification. Empirically, block verification provides modest but consistent wall-clock speedups over the standard token verification algorithm of 5%-8% in a wide range of tasks and datasets. Given that block verification does not increase code complexity, maintains the strong lossless guarantee of the standard speculative decoding verification algorithm, cannot deteriorate performance, and, in fact, consistently improves it, it can be used as a good default by speculative decoding implementations.

## 1. Introduction

Modern large language models (LLMs) (Chowdhery et al., 2022; Touvron et al., 2023; Achiam et al., 2023; Gemini Team et al., 2023) are often decoded through autoregressive sampling, where generating $k$ tokens requires $k$ costly serial evaluations of the model. To improve generation latency, Leviathan et al. (2022) proposed *speculative decoding*, which enables the LLM to generate several tokens concurrently. In each iteration, conditioned on the current decoded prefix, a guess of the next block of $\gamma$ tokens is made with a small computational cost (*e.g.,* generated from a smaller model or a heuristic). Each of the resulting $\gamma + 1$ prefixes are then evaluated by the large model in parallel. To guarantee that the final output follows the same distribution as that of the large model, some of the generated tokens are accepted while others are rejected. The accepted tokens, and an extra token sampled from a residual distribution, are then added to the prefix to start the next iteration, until generation ends. See Figure 1 for a diagram of the algorithm and Algorithm 3 for a detailed presentation.

In (Leviathan et al., 2022), the drafts are verified through a sequence of token-level rejection steps. More specifically, given prefix $\boldsymbol{c}$, let $X_1, X_2, \dots, X_\gamma$ be one sample block of length $\gamma$ from a small model $\mathcal{M}_s$, where $\forall i \le \gamma$, $X_i \sim \mathcal{M}_s(\cdot \mid \boldsymbol{c}, X_{\le i-1})$. Using the conditional distributions under the target large model $\mathcal{M}_b$ returned by the parallel evaluation step $(\forall i \le \gamma, \mathcal{M}_b(\cdot \mid \boldsymbol{c}, X_{\le i-1}))$ , the algorithm goes through draft tokens sequentially, and accepts each token $X_i$ with probability

$$\min\left\{1, \frac{\mathcal{M}_b(X_i \mid \boldsymbol{c}, X^{i-1})}{\mathcal{M}_s(X_i \mid \boldsymbol{c}, X^{i-1})}\right\}, \tag{1}$$

where $\mathcal{M}_b$ is the desired large model. The process continues until a token is rejected, and an extra token is sampled, for free, according to a residual distribution (see Algorithm 1 and (Leviathan et al., 2022) for more details). We refer to this algorithm as *Token Verification*.

Since its introduction (Leviathan et al., 2022), and despite many follow up works (*e.g.,* better drafting mechanisms or more draft blocks), all existing algorithms use the same token-by-token verification procedure. See Appendix A for a more detailed discussion of related works.

In this work, we make the surprising observation that *the standard token verification algorithm, is not optimal*, and propose a *strictly better method*. Our key observation is that we can increase the number of accepted tokens, while maintaining the identical distribution guarantee, by *jointly verifying the entire block of draft tokens* instead of verifying each token independently.

[1]Google Research. Authors marked by † are listed alphabetically. Correspondence to: Ziteng Sun <zitengsun@google.com>.

*Work to be presented at the ES-FoMo Workshop at ICML 2024*, Vienna, Austria. Copyright 2024 by the author(s).

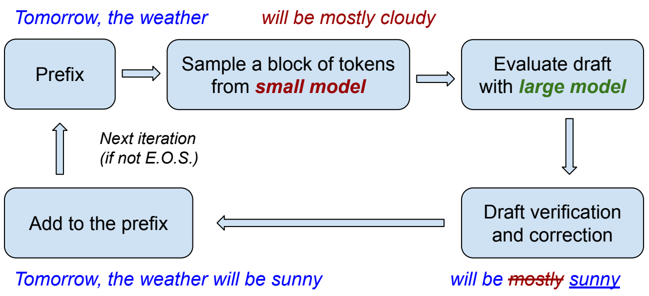
Tomorrow, the weather
will be mostly cloudy
Prefix
Sample a block of tokens from small model
Evaluate draft with large model
Next iteration (if not E.O.S.)
Add to the prefix
Draft verification and correction
Tomorrow, the weather will be sunny
will be ~~mostly~~ sunny

*Figure 1.* One iteration of speculative decoding (Algorithm 3). The prefixes are depicted in blue. The draft drawn from the small model is shown in red, and the verified draft tokens are then marked in blue and the underline is used to depict tokens obtained from the residual distribution.

Our proposed algorithm, which we call *Block Verification*, has the following advantages:

- **Simple to use.** The algorithm is a plug-and-play replacement of the standard token verification algorithm of speculative decoding. It does not incur additional computation or code complexity costs.
- **Strict improvement.** With the same drafting model, the speedup of block verification is no worse than that of the standard token verification. Moreover, we show that block verification is an optimal verification procedure (Theorem 2).
- **Identical distribution.** Importantly, our method is not an approximation and maintains the identical distribution guarantee of speculative decoding.

We empirically test *block verification* and compare it with the standard *token verification* on a wide range of tasks and datasets. We show that our algorithm consistently improves over block efficiency (i.e. the expected number of generated tokens) by 7%-10% and overall empirical wall clock times by 5%-8% (see Table 1). To the best of our knowledge, our algorithm is the *first* to provide improvements only through the verification phase of speculative decoding. The improvements can be combined with improvements obtained from other works that aim at improving the drafting phase. Since these improvements come for free, our block verification algorithm can be used as the draft verification algorithm by default by all speculative decoding implementations.

## 2. A Motivating Example

The standard token verification algorithm stochastically rejects draft tokens with a higher probability from $\mathcal{M}_s$ than from $\mathcal{M}_b$. This is necessary to guarantee that the generated tokens follow the same distribution as that of $\mathcal{M}_b$. Our main observation is that considering whether to reject a block of draft tokens jointly, instead of one-by-one, can result in accepting more tokens. We now illustrate this through a simple example.

Consider the following trivial language model whose token space consists only of 2 tokens: $a$ and $b$. Further, assume that both the large model $\mathcal{M}_b$ and the small model $\mathcal{M}_s$ are context-independent, and specifically that $\forall \boldsymbol{c}$,

$$\mathcal{M}_b(a) = 1/3, \qquad \mathcal{M}_b(b) = 2/3.$$
$$\mathcal{M}_s(a) = 2/3, \qquad \mathcal{M}_s(b) = 1/3.$$

In this setting, with a block size of $\gamma = 2$, since the total variation (TV) distance $d_{\text{TV}}(\mathcal{M}_b, \mathcal{M}_s) = 1/3$, the expected number of accepted tokens[1] from $\mathcal{M}_s$ with the token verification algorithm is $1 - 1/3 + (1 - 1/3)^2 = 10/9$ (see analysis in (Leviathan et al., 2022; Sun et al., 2023)).

**An ideal algorithm with full information.** Suppose an algorithm can decide on what tokens to accept from $\mathcal{M}_s$ based on the *full joint distributions* of both tokens, *i.e.,*

$$\mathcal{M}_b(aa) = 1/9, \qquad \mathcal{M}_b(ab) = 2/9,$$
$$\mathcal{M}_b(ba) = 2/9, \qquad \mathcal{M}_b(bb) = 4/9.$$
$$\mathcal{M}_s(aa) = 4/9, \qquad \mathcal{M}_s(ab) = 2/9,$$
$$\mathcal{M}_s(ba) = 2/9, \qquad \mathcal{M}_s(bb) = 1/9.$$

The algorithm would have performed the following improved acceptance logic: always accept $X_1X_2$ when $X_1X_2 = ab$, $ba$, or $bb$ since $\mathcal{M}_b(X_1X_2) \geq \mathcal{M}_s(X_1X_2)$, and accept $aa$ with probability $\mathcal{M}_b(aa)/\mathcal{M}_s(aa) = 1/4$ (correcting the samples to $bb$). The expected number of accepted tokens from $\mathcal{M}_s$ now becomes: $2(\mathcal{M}_s(ab) + \mathcal{M}_s(ba) + \mathcal{M}_s(bb) + 1/4 \times \mathcal{M}_s(aa)) = 12/9$. This illustrates the benefit of considering the distribution of draft blocks jointly.

**Verification with partial information.** In general the full distribution over the next block of tokens is intractable to calculate. Instead, we only have access to the conditional distributions of the next token along the *sample path* of

[1] This is different from the number of generated token in one iteration, which is the number of accepted tokens plus one (corrected token).

the draft block, $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i), \mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)$ for various $i$'s. To emphasize, the ideal rejection logic does not need access to the full distribution, but care is needed in properly assigning the residual distribution. Our block verification does exactly this, as follows.

With block verification, when the draft tokens $X_1X_2 = ab$ or $bb$, $\Pr(\text{Accept } X_1X_2) = 1$ similar to the idealized algorithm. When $X_1X_2 = aa$, $\Pr(\text{Accept } X_1X_2) = 1/4$, and else the algorithm rejects both tokens and only corrects the first token to $b$ since the algorithm doesn't have access to $\mathcal{M}_b(\cdot \mid b)$. When $X_1X_2 = ba$, it always accepts $b$, and then accepts $a$ with probability $1/2$ (else it corrects the second token to $b$). Importantly, the marginal distributions of the generated tokens at the first token and the second token are always $\mathcal{M}_b(\cdot)$. We then simply add the generated tokens to the prefix and proceed to the next iteration. The expected number of accepted tokens is $2 \times (\mathcal{M}_s(ab) + \mathcal{M}_s(bb)) + (1 + 1/2) \times \mathcal{M}_s(ba) + 1/4 \times 2 \times \mathcal{M}_s(aa) = 11/9$, which is better than $10/9$ obtained by token verification. This example proves the following result:

**Lemma 1.** *The standard token verification algorithm of speculative decoding is not optimal.*

Note that while the expected number of accepted tokens in the example for block verification ($11/9$) is higher than that of the standard token verification algorithm ($10/9$), it is still less than that of the ideal algorithm with access to the full distribution ($12/9$). In Appendix B, we show that block verification is indeed optimal in the partial information case, with natural assumptions.

## 3. Block Verification

In this section, we extend the above intuition to develop a general block verification algorithm, which works for standard speculative decoding with partial information. The high-level idea is to couple the acceptance of each draft token with other draft tokens. To do this, the algorithm considers draft sub-blocks with different lengths, and decides whether to accept each sub-block independently. The final accepted draft block is the longest accepted sub-block in the above process. The acceptance probabilities for each sub-block and the residual distributions are carefully chosen to maintain the distribution guarantee of the final output, and achieve optimal speedup.

See Algorithm 2 for a sketch implementation of block verification, and Algorithm 1 for a sketch implementation of the standard token verification for comparison. Note that the implementations follow the same overall structure (the differences are highlighted). See Algorithm 3 for the outer loop of the speculative decoding algorithm, which remains unchanged for both verification methods. See Figure 2 for additional definitions. See Appendix C for sketch Python implementations.

Importantly, note that token verification stops as soon as a token is rejected (**break** in Line 9 of Algorithm 1), while block verification always operates on the full block.

**Algorithm 3** Speculative decoding (SPECDEC) (Leviathan et al., 2022)

```
Input: Prefix $\boldsymbol{c}$, large model $\mathcal{M}_b$, draft model $\mathcal{M}_s$. Draft
    length $\gamma$. Verification algorithm VERIFY.
1: while E.O.S $\notin (X^\tau, Y)$ do
2:     Sample $X_1, \ldots, X_\gamma \sim \mathcal{M}_s(\cdot \mid \boldsymbol{c})$ using autoregressive sampling, keep the conditional
       probabilities at each step $\mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)$ for $i = 0, \ldots, \gamma - 1$.   {Obtain draft block.}
3:     Call the large model $\mathcal{M}_b$ and compute conditional probabilities $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)$
       for $i = 0, 1, \ldots, \gamma$ in parallel.   {Parallel scoring.}
4:     Get the accepted tokens $(X^\tau, Y)$ with draft verification   {Draft verification and correction.}

       VERIFY$(X^\gamma, \{\mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma-1}, \{\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma})$.

5:     $\boldsymbol{c} \leftarrow \boldsymbol{c}, X^\tau, Y$.   {Add decoded tokens to the prefix.}
6: end while
```

**Theoretical guarantees.** Speculative decoding with block verification preserves the distribution of its outputs (Theorem 1). Moreover, block verification achieves the *optimal* speedup among all valid draft verification algorithms in speculative decoding (Algorithm 3), resulting in a strict improvement over the standard token verification (Theorem 2). We defer the formal statements to Appendix B. Below we give some intuitions on the parameter choices.

**Intuition on parameter choices.** The key quantities for achieving the theoretical guarantees are $p_i$'s, which correspond to the probability that the sub-block $X^i$ will be kept in the final output (see Lemma 3 in Appendix D.1). The per-step acceptance probability $h_i$ in Equation (4) is then decided based on $p_i$'s since block verification keeps the longest accepted sub-block.

*Intuition: $p_i$'s.* To start, we ignore the minimum operation in the recursion. Then each $p_i$ is simply $\mathcal{M}_b(X^i \mid \boldsymbol{c})/\mathcal{M}_s(X^i \mid \boldsymbol{c})$, which is an upper bound on the actual $p_i$'s. This guarantees that the probability of getting $X^i$ by accepting it from the draft will be at most $\mathcal{M}_b(X^i \mid \boldsymbol{c})$, and hence the algorithm is not accepting $X^i$ more than needed.

Moreover, the minimum operation also guarantees that for any prefix $X^j$ that could be obtained from multiple draft sample paths, the distribution over subsequent tokens are always the same $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^j)$. This enables block verifi-

**Algorithm 1** TOKENVERIFY

```
Input: Draft block $X^\gamma$; small model distributions $\forall i < \gamma, \mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)$; large model distributions $\forall i \le \gamma, \mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)$.
1: Sample $\eta_1, \ldots, \eta_\gamma \sim U(0,1)$.
2: Set $\tau = 0$.
3: for $i = 1, \ldots \gamma$ do
5:   Set $h_i = \min\{\frac{\mathcal{M}_b(X_i|\boldsymbol{c},X^{i-1})}{\mathcal{M}_s(X_i|\boldsymbol{c},X^{i-1})}, 1\}$.
6:   if $\eta_i \le h_i$ then
7:     Set $\tau = i$.
8:   else
9:     break.
10:  end if
11: end for
12: if $\tau = \gamma$ then
13:   Sample $Y$ from $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^\gamma)$.
14: else
15:   Sample $Y$ from $p^{\text{token}}_{\text{res}}(\cdot \mid \boldsymbol{c}, X^\tau)$ as in Equation (2).
16: end if
17: Return $X^\tau, Y$.
```

**Algorithm 2** BLOCKVERIFY

```
Input: Draft block $X^\gamma$; small model distributions $\forall i < \gamma, \mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)$; large model distributions $\forall i \le \gamma, \mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)$.
1: Sample $\eta_1, \ldots, \eta_\gamma \sim U(0,1)$.
2: Set $\tau = 0$, $p_0 = 1$.
3: for $i = 1, \ldots \gamma$ do
4:   Set $p_i = \min\{p_{i-1}\frac{\mathcal{M}_b(X_i|\boldsymbol{c},X^{i-1})}{\mathcal{M}_s(X_i|\boldsymbol{c},X^{i-1})}, 1\}$.
5:   Set $h_i$ as in Equation (4).
6:   if $\eta_i \le h_i$ then
7:     Set $\tau = i$.
8:   else
9:     continue.
10:  end if
11: end for
12: if $\tau = \gamma$ then
13:   Sample $Y$ from $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^\gamma)$.
14: else
15:   Sample $Y$ from $p^{\text{block}}_{\text{res}}(\cdot \mid \boldsymbol{c}, X^\tau)$ as in Equation (3).
16: end if
17: Return $X^\tau, Y$.
```

cation to be used as a plug-and-play replacement of token verification in speculative decoding (Algorithm 3).

*Intuition:* $p^{\text{block}}_{\text{res}}$. For any $X_i$, the probability that it is in the final accepted block is $\mathcal{M}_s(X^i)p_i(X^i)$. The goal of choosing $p^{\text{block}}_{\text{res}}$ is to make sure that the distribution on the next token follows $\mathcal{M}_b(\cdot \mid X^i)$. For any possible next token $x$, $(X^i, x)$ could also be obtained by accepting $X^{i+1}$ when $X_{i+1} = x$, with a probability of $\mathcal{M}_s(X^i, x)p_{i+1}(X^i, x)$, which should be subtracted to obtain the residual mass on $(X^i, x)$. This leads to the choice of $p^{\text{block}}_{\text{res}}$ in Equation (3).

## 4. Experiment Setup

We use PALM-2 models (Chowdhery et al., 2022) for the drafter and target models, with PALM-2-S as the large target model and PALM-2-XXS / PALM-2-XXXS as the small drafter model. The order of the sizes of the models is PALM-2-XXXS < PALM-2-XXS < PALM-2-S. We evaluate on prompts from a wide range of datasets and tasks, including Language modeling with one-billion language benchmark (LM1B) (Chelba et al., 2013), ChatGPT prompts sourced from LearnGPT (GPT Prompt) (Rashad, 2023), reasoning questions (WebQA) (Berant et al., 2013), physical commonsense reasoning questions (PIQA) (Bisk et al., 2020), scraped conversations with ChatGPT (ShareGPT) (Rashad, 2023; RyokoAI, 2023), summarization tasks (XSum) (Narayan et al., 2018), grade school math problems (GSM8K) (Cobbe et al., 2021), and German to English translation (WMT DeEn) (Bojar et al., 2014). For all datasets, we decode the first 1000 prompts using a max input prompt length of 512 and decode up to 128 output tokens.

## 5. Results

We empirically compare speculative decoding with block verification to speculative decoding with token verification, and find that block verification provides small yet consistent improvements in a wide range of settings, both when measuring idealized *block efficiency* and real world *wall clock time*.

*Block efficiency* measures the speedup in an idealized settings where we neglect the evaluation time of the draft model and assume that we have enough compute capacity for evaluating the large model in parallel. I.e. it measures the average number of decoded tokens per serial call to the target model. We observe consistent improvements for all datasets and draft models[2]. For $\gamma = 8$ with PALM-2-XXS as the drafter, the improvement in block efficiency ranges from 7.00% to

[2]The improvemetn is also consistent across different runs with the standard deviations much smaller than the mean, indicating statistical significance.

Residual distribution in Algorithm 1 (Line 15): $\forall x \in \mathcal{X}$,

$$p_{\text{res}}^{\text{token}}(x \mid \boldsymbol{c}, X^i) = \frac{\max\{\mathcal{M}_b(x \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x \mid \boldsymbol{c}, X^i), 0\}}{\sum_{x' \in \mathcal{X}} \mathcal{M}_b(x' \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x' \mid \boldsymbol{c}, X^i), 0\}}. \tag{2}$$

Residual distribution in Algorithm 2 (Line 15): $\forall x \in \mathcal{X}$,

$$p_{\text{res}}^{\text{block}}(x \mid \boldsymbol{c}, X^i) = \frac{\max\{ p_i \cdot \mathcal{M}_b(x \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x \mid \boldsymbol{c}, X^i), 0\}}{\sum_{x' \in \mathcal{X}} \max\{ p_i \cdot \mathcal{M}_b(x' \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x' \mid \boldsymbol{c}, X^i), 0\}}. \tag{3}$$

Acceptance probability in Algorithm 2 (Line 5): $h_\gamma = p_\gamma$, and when $i < \gamma$,

$$h_i = \frac{\sum_{x' \in \mathcal{X}} \max\{p_i \cdot \mathcal{M}_b(x' \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x' \mid \boldsymbol{c}, X^i), 0\}}{\sum_{x' \in \mathcal{X}} \max\{p_i \cdot \mathcal{M}_b(x' \mid \boldsymbol{c}, X^i) - \mathcal{M}_s(x' \mid \boldsymbol{c}, X^i), 0\} + 1 - p_i}. \tag{4}$$

*Figure 2.* Definitions of accept probabilities and residual distributions in Algorithms 1 and 2.

$10.06\%$ with an average of $8.30\%$.

We also observe consistent improvements in *wall clock time*, which measures the actual speedup, including all the real-world overheads. We refer the readers to prior work (Leviathan et al., 2022; Chen et al., 2023a) to a more detailed discussion of these overheads. For $\gamma = 8$ with PALM-2-XXS as the drafter, the improvement in block efficiency ranges from $5.36\%$ to $8.14\%$ with an average of $6.49\%$. The detailed numbers for this setting are listed in Table 1.

**The effect of draft length $\gamma$.** We also perform comparisons of the algorithms for other block lengths ($\gamma = 4$ and $\gamma = 8$) and observe consistent improvements. We plot the average improvement over all datasets in Figure 4. With the same drafter, the relative improvement of block verification over token verification increases as $\gamma$ increases. This is consistent with our intuition since when $\gamma = 1$, the two algorithms are the same and as $\gamma$ increases, block verification would benefit more from coordinating the acceptance rule considering the realization of all tokens in the draft block.

**The effect of the drafter.** We also consider the effect of the quality of the drafter on the improvement. In Figure 3, we list the average block efficiency and wall clock speed up under different draft lengths for both drafters. Note that PALM-2-XXS is a larger model than PALM-2-XXXS, and hence a better drafter in terms of quality, as demonstrated by the better average block efficiencies in the table. In Figure 4, we plot the average improvement under different drafter models, PALM-2-XXS and PALM-2-XXXS. The improvements hold for both drafters. And the relative improvement in block efficiency under PALM-2-XXS is greater than that under PALM-2-XXXS. This shows that the improvement obtained from block verification can be combined with the improvement on the quality of the drafter, and the improvement might be more significant under better drafters.

| $\gamma$ | Drafter | TOKENV | | BLOCKV | |
|---|---|---|---|---|---|
| | | BE | WS | BE | WS |
| 4 | XXS | 2.89 | 2.44 | 2.99 | 2.50 |
| | XXXS | 2.35 | 2.36 | 2.43 | 2.43 |
| 6 | XXS | 3.23 | 2.43 | 3.43 | **2.54** |
| | XXXS | 2.50 | 2.39 | 2.63 | 2.50 |
| 8 | XXS | 3.41 | 2.30 | **3.70** | 2.45 |
| | XXXS | 2.57 | 2.28 | 2.73 | 2.40 |

*Figure 3.* Average block efficiency (BE) and wall clock speedup (WS) across all datasets for token verification (TOKENV) and block verification (BLOCKV) with different $\gamma$. The large model is PALM-2-S and the drafter model is either PALM-2-XXS (XXS) or PALM-2-XXXS (XXXS).

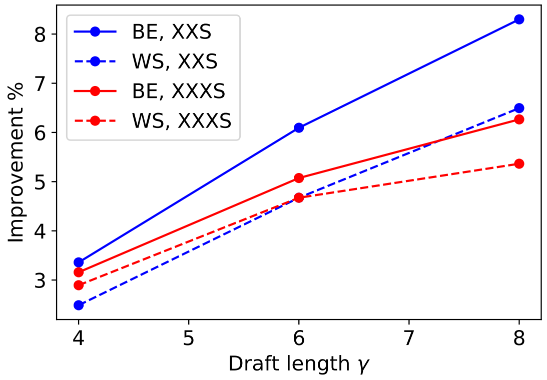

*Figure 4.* Average relative improvement of block verification over token verification in block efficiency (BE) and wall clock speedup (WS) across all datasets for different drafters and draft lengths.

*Table 1.* Speedup comparison between token verification (TOKENV) and block verification (BLOCKV) with $\gamma = 8$ and PALM-2-XXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TOKENV | BLOCKV | Improve. ↑ % | TOKENV | BLOCKV | Improve. ↑ % |
| LM1B | $3.21 \pm 0.01$ | $3.49 \pm 0.02$ | $8.68 \pm 0.79$ | $2.17 \pm 0.01$ | $2.32 \pm 0.01$ | $6.85 \pm 0.74$ |
| GPT Prompt | $3.41 \pm 0.04$ | $3.76 \pm 0.02$ | $10.06 \pm 1.66$ | $2.30 \pm 0.02$ | $2.48 \pm 0.01$ | $8.14 \pm 1.55$ |
| WebQA | $3.44 \pm 0.01$ | $3.70 \pm 0.01$ | $7.53 \pm 0.24$ | $2.32 \pm 0.00$ | $2.45 \pm 0.01$ | $5.75 \pm 0.22$ |
| PIQA | $3.40 \pm 0.02$ | $3.68 \pm 0.00$ | $8.30 \pm 0.62$ | $2.29 \pm 0.01$ | $2.44 \pm 0.00$ | $6.52 \pm 0.58$ |
| ShareGPT | $3.34 \pm 0.01$ | $3.62 \pm 0.03$ | $8.45 \pm 0.98$ | $2.25 \pm 0.01$ | $2.40 \pm 0.02$ | $6.68 \pm 0.91$ |
| XSum | $3.49 \pm 0.02$ | $3.76 \pm 0.01$ | $7.63 \pm 0.94$ | $2.35 \pm 0.01$ | $2.49 \pm 0.01$ | $5.82 \pm 0.88$ |
| GSM8K | $3.81 \pm 0.01$ | $4.15 \pm 0.03$ | $8.74 \pm 0.56$ | $2.55 \pm 0.01$ | $2.73 \pm 0.02$ | $6.84 \pm 0.51$ |
| WMT-DeEn | $3.19 \pm 0.01$ | $3.41 \pm 0.02$ | $7.00 \pm 0.78$ | $2.15 \pm 0.01$ | $2.27 \pm 0.01$ | $5.36 \pm 0.73$ |
| Average | 3.41 | 3.70 | 8.30 | 2.30 | 2.45 | 6.49 |

The detailed numbers on experiments performed with different drafters, different datasets, and different draft lengths are listed in Appendix E.

## References

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

J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In D. Yarowsky, T. Baldwin, A. Korhonen, K. Livescu, and S. Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160.

Y. Bisk, R. Zellers, R. Le bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. s. Tamchyna. 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, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Z. Chen, X. Yang, J. Lin, C. Sun, J. Huang, and K. C.-C. Chang. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv:2312.11462*, 2023b.

Z. Chen, A. May, R. Svirschevski, Y. Huang, M. Ryabinin, Z. Jia, and B. Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding, 2024.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

M. Elhoushi, A. Shrivastava, D. Liskovich, B. Hosmer, B. Wasti, L. Lai, A. Mahmoud, B. Acun, S. Agarwal, A. Roman, A. A. Aly, B. Chen, and C.-J. Wu. Layer-skip: Enabling early exit inference and self-speculative decoding, 2024.

Gemini Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction, 2024.

Z. He, Z. Zhong, T. Cai, J. D. Lee, and D. He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2022.

Y. Li, F. Wei, C. Zhang, and H. Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty, 2024.

X. Liu, L. Hu, P. Bailis, I. Stoica, Z. Deng, A. Cheung, and H. Zhang. Online speculative decoding, 2023.

X. Miao, G. Oliaro, Z. Zhang, X. Cheng, Z. Wang, R. Y. Y. Wong, Z. Chen, D. Arfeen, R. Abhyankar, and Z. Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

S. Narayan, S. B. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

J. P. Quirk and R. Saposnik. Admissibility and measurable utility functions. *The Review of Economic Studies*, 29(2): 140–146, 1962.

M. Rashad. Chatgpt-prompts. `https://huggingface.co/datasets/MohamedRashad/ChatGPT-prompts`, 2023.

RyokoAI. Sharegpt. `https://huggingface.co/datasets/RyokoAI/ShareGPT52K`, 2023.

M. Stern, N. Shazeer, and J. Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

H. Sun, Z. Chen, X. Yang, Y. Tian, and B. Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding, 2024.

Z. Sun, A. T. Suresh, J. H. Ro, A. Beirami, H. Jain, and F. Yu. Spectr: Fast speculative decoding via optimal transport. *arXiv preprint arXiv:2310.15141*, 2023.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

H. Xia, Z. Yang, Q. Dong, P. Wang, Y. Li, T. Ge, T. Liu, W. Li, and Z. Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

N. Yang, T. Ge, L. Wang, B. Jiao, D. Jiang, L. Yang, R. Majumder, and F. Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023.

J. Zhang, J. Wang, H. Li, L. Shou, K. Chen, G. Chen, and S. Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding, 2024.

Y. Zhou, K. Lyu, A. S. Rawat, A. K. Menon, A. Rostamizadeh, S. Kumar, J.-F. Kagy, and R. Agarwal. Distillspec: Improving speculative decoding via knowledge distillation, 2023.

## A. Related work

**Parallel decoding.** Our work improves speculative decoding (Leviathan et al., 2022), a framework for decoding several tokens concurrently. *Draft and verify* (Stern et al., 2018) was an earlier work, which proposed to independently predict and decode several tokens in parallel, for the greedy decoding case (zero temperature). Speculative decoding has later also been proposed in (Chen et al., 2023a).

**Improving speculative decoding.** There have been many works aiming to improve speculative decoding. In Table 2, we list a set of works in the draft and verify framework with a breakdown of their drafting and verification algorithms. See (Xia et al., 2024) for a comprehensive study. In the single-draft case, despite several works have worked on improving the drafting phase of speculative decoding (He et al., 2023; Chen et al., 2023b; Sun et al., 2024; Zhou et al., 2023; Liu et al., 2023; Gloeckle et al., 2024; Zhang et al., 2024; Elhoushi et al., 2024). However, all existing algorithms use the token verification algorithm, and little has been done to improve the verification phase. Our proposed block verification algorithm can be used in tandem with the drafting techniques in Table 2, yielding combined gains. We leave a more systematic study of the improvement of block verification in these cases for future study.

**Multiple drafts.** Recently, speculative decoding is extended to multiple drafts (Sun et al., 2023; Miao et al., 2023) and new verification algorithms for the multi-draft scenario are proposed (Li et al., 2024; Chen et al., 2024). While increasing the number of draft sequences has shown to improve the overall speedup, it comes at the cost of more computation. Their verification algorithm is a generalized token verification procedure. We leave extending the block verification idea to the multi-sample case as an interesting future direction.

*Table 2.* A summary of recent works on the draft and verify framework. Our work is the first to introduce block level draft verification. Temperature 0 refers to greedy decoding and non-zero temperature refers to sampling.

| Work | # drafts | Temp. | Drafting | Verification |
|---|---|---|---|---|
| (Stern et al., 2018) | 1 | 0 | parallel softmax layers | token matching |
| (Yang et al., 2023) | 1 | 0 | additional text | token matching |
| (Leviathan et al., 2022) | 1 | $\geq 0$ | small LM | TOKENVERIFY (Algorithm 1) |
| (Chen et al., 2023a) | 1 | $\geq 0$ | small LM | TOKENVERIFY (Algorithm 1) |
| (He et al., 2023) | 1 | $\geq 0$ | database retrieval | TOKENVERIFY (Algorithm 1) |
| (Chen et al., 2023b) | 1 | $\geq 0$ | cascade of small LMs | TOKENVERIFY (Algorithm 1) |
| (Sun et al., 2024) | 1 | $\geq 0$ | hierarchical drafters | TOKENVERIFY (Algorithm 1) |
| (Zhou et al., 2023) | 1 | $\geq 0$ | distilled small LMs | TOKENVERIFY (Algorithm 1) |
| (Liu et al., 2023) | 1 | $\geq 0$ | distilled small LMs | TOKENVERIFY (Algorithm 1) |
| (Gloeckle et al., 2024) | 1 | $\geq 0$ | parallel softmax layers | TOKENVERIFY (Algorithm 1) |
| (Zhang et al., 2024) | 1 | $\geq 0$ | layer skip | TOKENVERIFY (Algorithm 1) |
| (Elhoushi et al., 2024) | 1 | $\geq 0$ | early exit | TOKENVERIFY (Algorithm 1) |
| **This work** | 1 | $\geq 0$ | small LM | BLOCKVERIFY (Algorithm 2) |
| (Sun et al., 2023) | $\geq 2$ | $\geq 0$ | small LM | SpecTr |
| (Miao et al., 2023) | $\geq 2$ | $\geq 0$ | small LM | multi-round TOKENVERIFY |
| (Li et al., 2024) | $\geq 2$ | $\geq 0$ | small LM | multi-round TOKENVERIFY |
| (Chen et al., 2024) | $\geq 2$ | $\geq 0$ | small LM | multi-round TOKENVERIFY |

## B. Theoretical Guarantees

In this section, we present the formal theoretical guarantees of block verification. Notably, that it produces the correct distribution and that it is optimal in terms of the expected number of generated tokens. Let $\mathcal{M}^*(\cdot \mid \boldsymbol{c})$ denote the distribution of the sequence up to the end of the generative process under model $\mathcal{M}$ and context $\boldsymbol{c}$.

**Definition 1** (Valid draft verification algorithm.)**.** An algorithm VERIFY is said to be a valid draft verification algorithm if $\forall \boldsymbol{c}$, models $\mathcal{M}_s, \mathcal{M}_b$, and block length $\gamma$, the outputs of Algorithm 3 (SPECDEC) with verification algorithm VERIFY satisfy

$$\textsc{SpecDec}(\boldsymbol{c}, \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify}) \sim_{\mathrm{p}} \mathcal{M}_b^*(\cdot \mid \boldsymbol{c})^{3}, \tag{5}$$

*i.e.,* it preserves the distribution of the output.

We now claim the following:

**Theorem 1.** *Block verification is a valid draft verification algorithm.*

I.e. speculative decoding with block verification preserves the distribution of the output sequence.

We now further claim that block verification is optimal for all valid draft verification algorithms.

**Theorem 2.** *For $i > 0$, let $N(i)$ be the number of decoded tokens after $i$ iterations in Algorithm 3. For any valid draft verification algorithm* $\textsc{Verify}$ *in Definition 1, we have $\forall \boldsymbol{c}, \mathcal{M}_s, \mathcal{M}_b$, $\gamma$, and $i$,*

$$\mathbb{E}_{\textsc{BlockVerify}}[N(i)] \geq \mathbb{E}_{\textsc{Verify}}[N(i)],$$

*where the randomness is over the randomness of the draft block and the randomness of the algorithm.*

*In particular,*

$$\mathbb{E}_{\textsc{BlockVerify}}[N(i)] \geq \mathbb{E}_{\textsc{TokenVerify}}[N(i)].$$

I.e., among all valid verification algorithms, speculative decoding with block verification decodes the most number of tokens in expectation in a fixed number of iterations. Note that since the computation overhead added by block verification is negligibly small, this establishes the overall optimality of the block verification algorithm. In particular, block verification provides a greater speedup than the standard token verification. We defer the proofs to Appendix D.

## C. Python Implementation

In this section we provide a sketch implementation of Block Verification (Algorithm 2) in Python. Note that these are meant for illustration purposes only and is not optimized for production settings. Let $V = |\mathcal{X}|$ be the size of the vocabulary.

The inputs to the algorithm are:

- `ps`: an $(\gamma + 1) \times V$ `numpy` array with the distributions from the large model $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)$.
- `qs`: an $\gamma \times V$ `numpy` array with the distributions from the draft model $\mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)$;
- `drafts`: a length-$\gamma$ `numpy` array with the ids of the draft tokens $X^\gamma$;

```python
def block_verification(
  ps: np.ndarray, qs: np.ndarray, drafts: np.ndarray) -> list[int]:
  draft_length, vocab_size = qs.shape
  qs.resize((draft_length+1, vocab_size))
  token_sequence = None # Will include the token sequence we return
  accept_probability = 1.0 # Acceptance probability for each sub-block
  probability_ratios = ps / qs
  # Add one token to indicate rejecting the sequence
  vocab_plus_one = np.arange(vocab_size + 1)
  for token_index, token_value in enumerate(xs):
    # Unnormalized residual probability
    sampling_weights[:vocab_size] = np.maximum(
        0, ps[token_index] * accept_probability - qs[token_index])
    # Unnormalized probability of rejecting the sequence
    sampling_weights[vocab_size] = 1 - accept_probability
    sampling_weights /= np.sum(sampling_weights)
    chosen_token = np.random.choice(vocab_plus_one, p=sampling_weights)
```

[3]We use $\sim_{\mathrm{p}}$ to denote that two distributions are the same.

```
    # Update the sequence
    if chosen_token < vocab_size:
      token_sequence = xs[:token_index] + [chosen_token]
    # Update the acceptance probability
    accept_probability = min(
        1, probability_ratios[token_index, token_value] * accept_probability)
  return token_sequence
```

For reference, here is a sketch implementation of the token verification algorithm (Algorithm 1):

```
def token_verification(
  ps: np.ndarray, qs: np.ndarray, drafts: np.ndarray) -> list[int]:
  draft_length, vocab_size = qs.shape
  qs.resize((draft_length+1, vocab_size))
  token_sequence = [] # Will include the token sequence we return
  probability_ratios = ps / qs
  token_index = 0
  vocab_range = np.arange(vocab_size)
  for token_value in xs:
    accept_probability = probability_ratios[token_index, token_value]
    if (not np.isfinite(accept_probability) or
        np.random.random() > accept_probability): # Rejection
      break
    token_index += 1
    token_sequence.append(token_value)
  # Calculate the residual distribution
  sampling_weights = np.maximum(0, ps[token_index] - qs[token_index])
  sampling_weights /= np.sum(sampling_weights)
  token_sequence.append(np.random.choice(vocab_range, p=sampling_weights))
  return token_sequence
```

## D. Formal Proofs

We start by setting up a few necessary notations. Let $\mathcal{X}$ be the space of output tokens. For $\ell > 1$, we use $\mathcal{M}^\ell(\cdot \mid \boldsymbol{c})$ to denote the joint distribution of the next $\ell$ tokens conditioned on the prefix under $\mathcal{M}$, *i.e.,* for all $x_1, \dots x_\ell \in \mathcal{X}^\ell$, $\mathcal{M}^\ell(x_1, \dots, x_\ell \mid \boldsymbol{c}) = \prod_{i=1}^{\ell} \mathcal{M}(x_i \mid \boldsymbol{c}, x^{i-1})$. We use $\mathcal{M}^*(\cdot \mid \boldsymbol{c})$ to denote the distribution of the sequence up to the end of the generative process. We use $\sim_{\mathrm{p}}$ to denote that two distributions are the same. Below we first describe a necessary and sufficient condition for a valid draft verification algorithm in Algorithm 3.

**Lemma 2.** *$\forall \boldsymbol{c}, \mathcal{M}_s, \mathcal{M}_b, \gamma$, let $X^\gamma$ be generated from $\mathcal{M}_s^\gamma(\cdot \mid \boldsymbol{c})$, and*

$$X^\tau, Y = \textsc{Verify}(X^\gamma, \{\mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma-1}, \{\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma}).$$

*Let $Z^{\gamma-\tau}$ be generated from $\mathcal{M}_b^{\gamma-\tau}(\cdot \mid \boldsymbol{c}, X^\tau, Y)$.*

Verify *is a valid draft verification algorithm (Definition 1) if and only if $\forall \boldsymbol{c}, \mathcal{M}_s, \mathcal{M}_b, \gamma$,*

$$X^\tau, Y, Z^{\gamma-\tau} \sim_{\mathrm{p}} \mathcal{M}_b^{\gamma+1}(\cdot \mid \boldsymbol{c}). \tag{6}$$

*Proof.* We first prove the forward direction by induction on the maximum generation length of $\mathcal{M}_b(\cdot \mid \boldsymbol{c})$. When the maximum generation length is 0, for all new context $\boldsymbol{c}'$, we have the next token is a point mass over E.O.S, *i.e.,*

$$\mathcal{M}_b(x \mid \boldsymbol{c}, \boldsymbol{c}') = \delta\{x = \text{E.O.S}\}.$$

Then Equation (6) implies that $\textsc{Verify}$ will only output E.O.S, which is the same as Definition 1. Suppose Equation (5) holds for all context and $\mathcal{M}_b$ with generation length at most $T$, for a context $\boldsymbol{c}$ and $\mathcal{M}_b$ with maximum generation length at most $T+1$, we have that the output of $\textsc{SpecDec}(\boldsymbol{c}, \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})$ is

$$X^\tau, Y, \textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify}).$$

Let $Z^{\gamma-\tau}$ be the first $\gamma - \tau$ tokens from $\textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})$, and $O^*$ be the tokens after. Since $X^\tau, Y$ is at least of length one, the generation length of $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^\tau, Y)$ is at most $T$. By the induction hypothesis, we have

$$Z^{\gamma-\tau} \sim_{\mathrm{p}} \mathcal{M}_b^{\gamma-\tau}(\cdot \mid \boldsymbol{c}, X^\tau, Y),$$

and

$$O^* \sim_{\mathrm{p}} \mathcal{M}_b^*(\cdot \mid \boldsymbol{c}, X^\tau, Y, Z^{\gamma-\tau}).$$

And hence by Equation (6),

$$\begin{aligned} \textsc{SpecDec}(\boldsymbol{c}, \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify}) &= X^\tau, Y, \textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify}) \\ &= X^\tau, Y, Z^{\gamma-\tau}, O^* \\ &\sim_{\mathrm{p}} \mathcal{M}_b^*(\cdot \mid \boldsymbol{c}). \end{aligned}$$

This completes the proof for the forward direction.

For the backward direction, we have Equation (5) implies that for all $X^\tau, Y$,

$$\textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})[: \gamma - \tau]^{4} \sim_{\mathrm{p}} \mathcal{M}_b^{\gamma-\tau}(\cdot \mid \boldsymbol{c}, X^\tau, Y).$$

Let $Z^{\gamma-\tau}$ be a draw from $\mathcal{M}_b^{\gamma-\tau}(\cdot \mid \boldsymbol{c}, X^\tau, Y)$, then

$$Z^{\gamma-\tau} \sim_{\mathrm{p}} \textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})[: \gamma - \tau].$$

And hence when $X^\tau, Y$ is the output of $\textsc{Verify}$,

$$\begin{aligned} X^\tau, Y, Z^{\gamma-\tau} &\sim_{\mathrm{p}} X^\tau, Y, \textsc{SpecDec}((\boldsymbol{c}, X^\tau, Y), \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})[: \gamma - \tau] \\ &\sim_{\mathrm{p}} \textsc{SpecDec}(\boldsymbol{c}, \mathcal{M}_b, \mathcal{M}_s, \gamma, \textsc{Verify})[: \gamma + 1] \\ &\sim_{\mathrm{p}} \mathcal{M}_b^{\gamma+1}(\cdot \mid \boldsymbol{c}). \end{aligned}$$

□

In all proofs below, we fix the context $\boldsymbol{c}$, and the models $\mathcal{M}_s$ and $\mathcal{M}_b$. We note that the proofs won't use specific information about these choices and hence can be easily extended to all cases.

## D.1. Proof of Theorem 1

By Lemma 2, it would be enough to prove that $\textsc{BlockVerify}$ satisfies Equation (6). For simplicity, we often refer to the sequence $(X^\tau, Y, Z^{\gamma-\tau})$ by $O^{\gamma+1}$. Since $O_{\gamma+1} \sim \mathcal{M}_b(\cdot \mid \boldsymbol{c}, O^\gamma)$ always holds, it is enough to prove the following

$$\forall \ell \leq \gamma, \forall x^\ell \in \mathcal{X}^\ell, \quad \Pr\left(O^\ell = x^\ell\right) = \mathcal{M}_b^\ell(x^\ell \mid \boldsymbol{c}), \tag{7}$$

Note that in $\textsc{BlockVerify}$, $p_i$'s depend on the draft tokens $X^\gamma$. The following definition makes this explicit. Let $p_i$ be such that $p_0 = 1$, and $\forall 1 \leq i \leq \gamma$, $x^i \in \mathcal{X}^i$,

$$p_i(x^i \mid \boldsymbol{c}) = \min\left\{ p_{i-1}(x^i \mid \boldsymbol{c}) \frac{\mathcal{M}_b(x_i \mid \boldsymbol{c}, x^{i-1})}{\mathcal{M}_s(x_i \mid \boldsymbol{c}, x^{i-1})}, 1 \right\}. \tag{8}$$

For most cases, when the prefix $\boldsymbol{c}$ is clear, we will ignore $\boldsymbol{c}$ and simply use $p_i(x^i) = p_i(x^i \mid \boldsymbol{c})$. We will only make the prefix explicit when necessary.

We first state the following lemma on the output distribution of $\textsc{BlockVerify}$.

[4] We use $\boldsymbol{v}[i : j]$ to denote the entries $i$ to $j$ in $\boldsymbol{v}$.

**Lemma 3.** *Let $X^\gamma \sim \mathcal{M}_s^\gamma(\cdot \mid \boldsymbol{c})$, and*

$$X^\tau, Y = \textsc{BlockVerify}(X^\gamma, \{\mathcal{M}_s(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma-1}, \{\mathcal{M}_b(\cdot \mid \boldsymbol{c}, X^i)\}_{i=0}^{\gamma}).$$

*Then we have $\forall i \leq \gamma$, and $x^i \in \mathcal{X}^i$,*

$$\Pr\left(\tau \geq \ell \mid X^i = x^i\right) = p_i(x^i).$$

We first prove Theorem 1 based on Lemma 3 and defer the proof of the lemma to Appendix D.3. We prove Equation (7) by induction on the time index $\ell$. When $\ell = 1$, $O_1$ is either $X_1$, or a residual sample from $p_{\text{res}}^{\text{block}}(\cdot \mid \boldsymbol{c})$ where

$$p_{\text{res}}^{\text{block}}(\cdot \mid \boldsymbol{c}) = \frac{\max\{\mathcal{M}_b(x \mid \boldsymbol{c}) - \mathcal{M}_s(x \mid \boldsymbol{c}), 0\}}{\sum_{x'} \max\{\mathcal{M}_b(x' \mid \boldsymbol{c}) - \mathcal{M}_s(x' \mid \boldsymbol{c}), 0\}},$$

Hence we have $\forall x \in \mathcal{X}$, by Lemma 3,

$$\begin{aligned}
&\Pr(O_1 = x) \\
&= \Pr(O_1 = x, \tau \geq 1) + \Pr(O_1 = x, \tau = 0) \\
&= \Pr(X_1 = x)\Pr(\tau \geq 1 \mid X_1 = x) + \sum_{x'} \Pr(X_1 = x')(1 - \Pr(\tau \geq 1 \mid X_1 = x')) \cdot p_{\text{res}}^{\text{block}}(x \mid \boldsymbol{c}) \\
&= \mathcal{M}_s(x \mid \boldsymbol{c}) \cdot p_1(x) + \sum_{x'} \mathcal{M}_s(x' \mid \boldsymbol{c})(1 - p_1(x')) \cdot p_{\text{res}}^{\text{block}}(x \mid \boldsymbol{c}) \\
&= \min\{\mathcal{M}_b(x \mid \boldsymbol{c}), \mathcal{M}_s(x \mid \boldsymbol{c})\} + \sum_{x'} \max\{\mathcal{M}_b(x' \mid \boldsymbol{c}) - \mathcal{M}_s(x' \mid \boldsymbol{c}), 0\} \cdot p_{\text{res}}^{\text{block}}(x \mid \boldsymbol{c}) \\
&= \min\{\mathcal{M}_b(x \mid \boldsymbol{c}), \mathcal{M}_s(x \mid \boldsymbol{c})\} + \max\{\mathcal{M}_b(x \mid \boldsymbol{c}) - \mathcal{M}_s(x \mid \boldsymbol{c}), 0\} \\
&= \mathcal{M}_b(x \mid \boldsymbol{c}).
\end{aligned}$$

Hence the Equation (7) holds for $\ell = 1$. Suppose Equation (7) holds up to an arbitrary $\ell < \gamma$. For $\ell = \ell + 1$, we have $O_{\ell+1}$ is either equal to $X_{\ell+1}$ when $\tau \geq \ell + 1$, or a sample from $p_{\text{res}}^{\text{block}}(\cdot \mid \boldsymbol{c}, X^\ell)$ when $\tau = \ell$, or a sample from $\mathcal{M}_b(\cdot \mid \boldsymbol{c}, O^\ell)$ when $\tau < \ell$. Hence $\Pr\left(O^{\ell+1} = x^{\ell+1}\right)$ can be broken down below:

$$\begin{aligned}
&\Pr\left(O^{\ell+1} = x^{\ell+1}\right) = \\
&\Pr\left(O^{\ell+1} = x^{\ell+1}, \tau \geq \ell + 1\right) + \Pr\left(O^{\ell+1} = x^{\ell+1}, \tau = \ell\right) + \Pr\left(O^{\ell+1} = x^{\ell+1}, \tau < \ell\right)
\end{aligned} \tag{9}$$

For the first term ($\tau \geq \ell + 1$), we have

$$\begin{aligned}
&\Pr\left(O^{\ell+1} = x^{\ell+1}, \tau \geq \ell + 1\right) \\
&= \Pr\left(X^{\ell+1} = x^{\ell+1}\right) \cdot \Pr\left(\tau \geq \ell + 1 \mid X^{\ell+1} = x^{\ell+1}\right) \\
&= \mathcal{M}_s(x^{\ell+1} \mid \boldsymbol{c}) \cdot p_{\ell+1}(x^{\ell+1}) \\
&= \mathcal{M}_s(x^\ell \mid \boldsymbol{c}) \cdot \min\{p_\ell(x^\ell)\mathcal{M}_b(x_{\ell+1} \mid \boldsymbol{c}, x^\ell), \mathcal{M}_s(x_{\ell+1} \mid \boldsymbol{c}, x^\ell)\}.
\end{aligned} \tag{10}$$

For the second term ($\tau = \ell$), we have

$$\begin{aligned}
&\Pr\left(O^{\ell+1} = x^{\ell+1}, \tau = \ell\right) \\
&= \Pr\left(X^\ell = x^\ell\right) \cdot \Pr\left(\tau = \ell \mid X^\ell = x^\ell\right) \cdot \Pr\left(O_{\ell+1} = x_{\ell+1} \mid O^\ell = x^\ell, \tau = \ell\right) \\
&= \Pr\left(X^\ell = x^\ell\right) \cdot \Pr\left(\tau = \ell \mid X^\ell = x^\ell\right) \cdot p_{\text{res}}^{\text{block}}(x_{\ell+1} \mid \boldsymbol{c}, x^\ell)
\end{aligned}$$

Note that,

$$\begin{aligned}
&\Pr\left(\tau = \ell \mid X^\ell = x^\ell\right) \\
&= \Pr\left(\tau \ge \ell \mid X^\ell = x^\ell\right) - \sum_x \mathcal{M}_s(x \mid \boldsymbol{c}, x^\ell) \cdot \Pr\left(\tau \ge \ell + 1 \mid \boldsymbol{c}, X^{\ell+1} = x^\ell, x\right) \\
&= p_\ell(x^\ell) - \sum_x \mathcal{M}_s(x \mid \boldsymbol{c}, x^\ell) \cdot p_{\ell+1}(x^\ell, x) \\
&= p_\ell(x^\ell) - \sum_x \min\{p_\ell(x^\ell)\mathcal{M}_b(x \mid \boldsymbol{c}, x^\ell), \mathcal{M}_s(x \mid \boldsymbol{c}, x^\ell)\} \\
&= \sum_x \max\{p_\ell(x^\ell)\mathcal{M}_b(x \mid \boldsymbol{c}, x^\ell) - \mathcal{M}_s(x \mid \boldsymbol{c}, x^\ell), 0\}.
\end{aligned}$$

And hence

$$\begin{aligned}
&\Pr\left(O^{\ell+1} = x^{\ell+1}, \tau = \ell\right) \\
&= \mathcal{M}_s(x^\ell \mid \boldsymbol{c}) \cdot \max\{p_\ell(x^\ell)\mathcal{M}_b(x_{\ell+1} \mid \boldsymbol{c}, x^\ell) - \mathcal{M}_s(x_{\ell+1} \mid \boldsymbol{c}, x^\ell), 0\}.
\end{aligned} \tag{11}$$

For the third term ($\tau < \ell$), by induction, and the generation process of $O^{\gamma+1}$, we have

$$\begin{aligned}
\Pr\left(O^{\ell+1} = x^{\ell+1}, \tau < \ell\right) &= \Pr\left(O^\ell = x^\ell, \tau < \ell\right) \cdot \Pr\left(O_{\ell+1} = x_{\ell+1} \mid O^\ell = x^\ell, \tau < \ell\right) \\
&= \left(\Pr\left(O^\ell = x^\ell\right) - \Pr\left(O^\ell = x^\ell, \tau \ge \ell\right)\right) \cdot \mathcal{M}_b(x_{\ell+1} \mid \boldsymbol{c}, x^\ell) \\
&= \left(\mathcal{M}_b(x^\ell \mid \boldsymbol{c}) - \mathcal{M}_s(x^\ell \mid \boldsymbol{c}) p_\ell(x^\ell)\right) \cdot \mathcal{M}_b(x_{\ell+1} \mid \boldsymbol{c}, x^\ell)
\end{aligned} \tag{12}$$

Plugging Equations (10) to (12) into Equation (9), we get $\forall x^{\ell+1} \in \mathcal{X}^{\ell+1}$,

$$\Pr\left(O^{\ell+1} = x^{\ell+1}\right) = \mathcal{M}_b(x^{\ell+1} \mid \boldsymbol{c}),$$

completing the induction step and hence the proof of Equation (7) and Theorem 1.

### D.2. Proof of Theorem 2

We first state the following lemma, which shows that in one iteration, among all valid draft verification algorithms, BLOCKVERIFY accepts each subsequence with the highest probability.

**Lemma 4.** *For draft verification algorithms that satisfy the constraints in Lemma 2, we have $\forall i \le \gamma$, and $x^i \in \mathcal{X}^i$,*

$$\Pr\left(\tau \ge i \mid X^i = x^i\right) \le p_i(x^i).$$

We defer the proof of the lemma to Appendix D.4 and first prove Theorem 2 based on the lemma.

We start by breaking down the expected number of decoded tokens $\mathbb{E}_{\text{VERIFY}}[N(i)]$ into the distribution of $N(i)$ on different sample paths. Let $O^* = O_1, O_2, \ldots,$ be the complete output sequence from speculative decoding. We set all tokens after E.O.S to be E.O.S as well. Then we have

$$\mathbb{E}_{\text{VERIFY}}[N(i)] = \sum_{\ell=1}^{\infty} \Pr_{\text{VERIFY}}\left(N(i) \ge \ell\right) = \sum_{x^* \in \mathcal{X}^*} \sum_{\ell=1}^{\infty} \Pr_{\text{VERIFY}}\left(O^* = x^*, N(i) \ge \ell\right).$$

Hence it would be enough to prove the following.

**Lemma 5.** *For all draft verification algorithms that satisfy the constraints in Lemma 2, we have $\forall \boldsymbol{c}$, and output $x^*$,*

$$\Pr_{\text{VERIFY}}\left(O^* = x^*, N(i) \ge \ell \mid \boldsymbol{c}\right) \le \Pr_{\text{BLOCKVERIFY}}\left(O^* = x^*, N(i) \ge \ell \mid \boldsymbol{c}\right) \tag{13}$$

We prove the lemma by induction on the number of iterations $i$. When $i = 1$, we have that $N(i) = \tau + 1$, where $\tau$ is the number of accepted tokens. Hence by the guarantee in Lemma 2,

$$\begin{aligned}
&\Pr_{\textsc{Verify}}(O^* = x^*, N(1) \geq \ell \mid \boldsymbol{c}) \\
&= \Pr_{\textsc{Verify}}\left(O^\ell = x^\ell, N(1) \geq \ell \mid \boldsymbol{c}\right) \cdot \mathcal{M}_b^*(x^{\ell+1:*} \mid \boldsymbol{c}, x^\ell) && \triangleleft \text{ Equation (21)} \\
&= \Pr_{\textsc{Verify}}\left(O^\ell = x^\ell, \tau \geq \ell - 1 \mid \boldsymbol{c}\right) \cdot \mathcal{M}_b^*(x^{\ell+1:*} \mid \boldsymbol{c}, x^\ell) \\
&= \Pr_{\textsc{Verify}}\left(X^{\ell-1} = x^{\ell-1}, \tau \geq \ell - 1 \mid \boldsymbol{c}\right) \cdot \mathcal{M}_b(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot \mathcal{M}_b^*(x^{\ell+1:*} \mid \boldsymbol{c}, x^\ell) \\
&= \mathcal{M}_s(x^{\ell-1} \mid \boldsymbol{c})\mathcal{M}_b^*(x^{\ell:*} \mid \boldsymbol{c}, x^{\ell-1}) \Pr_{\textsc{Verify}}\left(\tau \geq \ell - 1 \mid X^{\ell-1} = x^{\ell-1}, \boldsymbol{c}\right) \\
&\leq \mathcal{M}_s(x^{\ell-1} \mid \boldsymbol{c})\mathcal{M}_b^*(x^{\ell:*} \mid \boldsymbol{c}, x^{\ell-1}) \Pr_{\textsc{BlockVerify}}\left(\tau \geq \ell - 1 \mid X^{\ell-1} = x^{\ell-1}, \boldsymbol{c}\right) && \triangleleft \text{ Lemmas 3 and 4} \\
&= \Pr_{\textsc{BlockVerify}}(O^* = x^*, N(1) \geq \ell \mid \boldsymbol{c}). && (14)
\end{aligned}$$

Suppose the lemma holds for all iterations up to $i$, for the $(i+1)$th iteration, let $\tau_i$ be the number of tokens accepted in the $(i+1)$th iteration, we have

$$\begin{aligned}
&\Pr_{\textsc{Verify}}(O^* = x^*, N(i+1) \geq \ell \mid \boldsymbol{c}) \\
&= \sum_{\ell' < \ell} \Pr_{\textsc{Verify}}(O^* = x^*, N(i) = \ell' \mid \boldsymbol{c}) \Pr_{\textsc{Verify}}(O^* = x^*, \tau_{i+1} \geq \ell - \ell' - 1 \mid \boldsymbol{c}, N(i) = \ell') \\
&= \Pr_{\textsc{Verify}}(O^* = x^* \mid \boldsymbol{c}) \sum_{\ell' < \ell} \Pr_{\textsc{Verify}}(N(i) = \ell' \mid O^* = x^*, \boldsymbol{c}) \\
&\qquad \cdot \Pr_{\textsc{Verify}}(O^* = x^*, \tau_{i+1} \geq \ell - \ell' - 1 \mid \boldsymbol{c}, N(i) = \ell') \\
&= \mathcal{M}_b(x^* \mid \boldsymbol{c}) \sum_{\ell' < \ell} \Pr_{\textsc{Verify}}(N(i) = \ell' \mid O^* = x^*, \boldsymbol{c}) \Pr_{\textsc{Verify}}(O^* = x^*, \tau_{i+1} \geq \ell - \ell' - 1 \mid \boldsymbol{c}, N(i) = \ell') && (15)
\end{aligned}$$

Let $\eta_{\textsc{Verify}}$ be a random variable distributed according to $\Pr_{\textsc{Verify}}(N(i) = \ell' \mid O^* = x^*, \boldsymbol{c})$, and

$$f_{\textsc{Verify}}(\eta) = \Pr_{\textsc{Verify}}(O^* = x^*, \tau_{i+1} \geq \ell - \eta - 1 \mid \boldsymbol{c}, N(i) = \eta).$$

Plugging these into Equation (15), we have

$$\Pr_{\textsc{Verify}}(O^* = x^*, N(i+1) \geq \ell \mid \boldsymbol{c}) = \mathcal{M}_b(x^* \mid \boldsymbol{c})\mathbb{E}_{\eta_{\textsc{Verify}}}\left[f_{\textsc{Verify}}(\eta)\right]$$

Note that let $\boldsymbol{c}' = \boldsymbol{c}, x^{\ell'}$, we have

$$\begin{aligned}
f_{\textsc{Verify}}(\eta) &= \Pr_{\textsc{Verify}}(O^* = x^*, \tau_{i+1} \geq \ell - \ell' - 1 \mid \boldsymbol{c}, N(i) = \ell') \\
&= \Pr_{\textsc{Verify}}\left(O^* = x^{\ell':*}, \tau \geq \ell - \ell' - 1 \mid \boldsymbol{c}'\right) \\
&\leq \Pr_{\textsc{BlockVerify}}\left(O^* = x^{\ell':*}, \tau \geq \ell - \ell' - 1 \mid \boldsymbol{c}'\right) \\
&= \Pr_{\textsc{BlockVerify}}(O^* = x^*, \tau_{i+1} \geq \ell - \ell' - 1 \mid \boldsymbol{c}, N(i) = \ell') \\
&= f_{\textsc{BlockVerify}}(\eta).
\end{aligned}$$

Hence we have

$$\Pr_{\textsc{Verify}}(O^* = x^*, N(i+1) \geq \ell \mid \boldsymbol{c}) \leq \mathbb{E}_{\eta_{\textsc{Verify}}}\left[f_{\textsc{BlockVerify}}(\eta)\right].$$

Note that $\Pr_{\textsc{BlockVerify}}(O^* = x^*, N(i+1) \geq \ell \mid \boldsymbol{c}) = \mathbb{E}_{\eta_{\textsc{BlockVerify}}}[f_{\textsc{BlockVerify}}(\eta)]$. It would be enough to prove that

$$\mathbb{E}_{\eta_{\textsc{Verify}}}\left[f_{\textsc{BlockVerify}}(\eta)\right] \leq \mathbb{E}_{\eta_{\textsc{BlockVerify}}}\left[f_{\textsc{BlockVerify}}(\eta)\right]. \tag{16}$$

By the induction hypothesis, we have $\eta_{\textsc{BlockVerify}}$ stochastically dominates (Quirk and Saposnik, 1962) $\eta_{\textsc{Verify}}$ for any valid verification algorithm. Moreover, by definition and Lemma 3,

$$f_{\textsc{BlockVerify}}(\eta) = \begin{cases} 0 & \text{if } \eta < \ell - \gamma - 1, \\ p_{\ell-\eta-1}(x^{\eta+1:\ell-1} \mid \boldsymbol{c}, x^{\eta}), & \text{if } \ell - \gamma - 1 \le \eta \le \ell - 1 \\ 1 & \text{if } \eta > \ell - 1, \end{cases}$$

which is an increasing function of $\eta$. To see this, for $\eta' = \eta + 1$, we can obtain $p_{\ell-\eta'-1}(x^{\eta'+1:\ell-1} \mid \boldsymbol{c}, x^{\eta'})$ by following the same recursion steps as in Equation (8) but replacing $p_1(x^{\eta+1:\ell-1} \mid \boldsymbol{c}, x^{\eta})$ with $p_0(x^{\eta+2:\ell-1} \mid \boldsymbol{c}, x^{\eta+1}) = 1$, and hence only increasing the values.

Hence Equation (16) holds (Quirk and Saposnik, 1962). This completes the induction step and the proof of Theorem 2.

## D.3. Proof of Lemma 3

Note that in Algorithm 2, $p_i = p_i(X^i)$. We prove the statement by backward induction. When $i = \gamma$, we have by definition of $h_\gamma^{\text{block}}$, $\forall x^\gamma \in \mathcal{X}^\gamma$,

$$\Pr\left(\tau \ge \gamma \mid X^\gamma = x^\gamma\right) = p_\gamma(x^\gamma).$$

Suppose the statement holds for $i \ge \ell$. When $i = \ell - 1$, we have

$$\begin{aligned} &\Pr\left(\tau \ge \ell - 1 \mid X^{\ell-1} = x^{\ell-1}\right) \\ &= \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot \Pr\left(\tau \ge \ell - 1 \mid X^\ell = x^\ell\right) \\ &= \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot \left(\Pr\left(\tau \ge \ell \mid X^\ell = x^\ell\right) + \Pr\left(\tau < \ell \mid X^\ell = x^\ell\right) \cdot h_\ell^{\text{block}}\right) \\ &= \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot \left(p_\ell(x^\ell) + (1 - p_\ell(x^\ell)) \cdot h_\ell^{\text{block}}\right), \\ &= \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot p_\ell(x^\ell) + h_\ell^{\text{block}} \cdot \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot (1 - p_\ell(x^\ell)). \end{aligned} \tag{17}$$

Note that in the definition of $h_\ell^{\text{block}}$ (Equation (4)),

$$\begin{aligned} &\sum_x \max\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}) - \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1}), 0\} \\ &= \sum_x \left(p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}) - \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1})\}\right) \\ &= p_{\ell-1}(x^{\ell-1}) - \sum_x \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1})\} \end{aligned}$$

Hence

$$h_\ell^{\text{block}} = \frac{p_{\ell-1}(x^{\ell-1}) - \sum_x \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1})\}}{1 - \sum_x \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1})\}}. \tag{18}$$

Moreover, we have

$$\begin{aligned} \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot p_\ell(x^\ell) &= \sum_{x_\ell \in \mathcal{X}} \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1}) \cdot p_\ell(x^\ell) \\ &= \sum_{x_\ell \in \mathcal{X}} \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x_\ell \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x_\ell \mid \boldsymbol{c}, x^{\ell-1})\} \\ &= \sum_{x \in \mathcal{X}} \min\{p_{\ell-1}(x^{\ell-1})\mathcal{M}_b(x \mid \boldsymbol{c}, x^{\ell-1}), \mathcal{M}_s(x \mid \boldsymbol{c}, x^{\ell-1})\}. \end{aligned} \tag{19}$$

Plugging (19) and (18) into (17), we get

$$\Pr\left(\tau \geq \ell - 1 \mid X^{\ell-1} = x^{\ell-1}\right) = p_{\ell-1}(x^{\ell-1}),$$

as desired. The lemma hence follows by induction.

## D.4. Proof of Lemma 4

Recall that we use $O^{\gamma+1}$ to denote the sequence $(X^{\tau}, Y, Z^{\gamma-\tau})$ in Equation (6). We prove the lemma by induction. When $i = 1$, the lemma holds since else the output probability of $x_1$ will be higher than $\mathcal{M}_b(x_1 \mid \boldsymbol{c})$. Suppose the lemma holds for all $i \leq \ell - 1$. When $i = \ell$, we prove that it also holds by contradiction. Suppose there exists $x^\ell \in \mathcal{X}^\ell$ such that

$$\Pr\left(\tau \geq \ell \mid X^{\ell} = x^{\ell}\right) > p_{\ell}(x^{\ell}). \tag{20}$$

If $\forall i \leq \ell - 1$, it satisfies that $p_i(x^i)\mathcal{M}_b(x_{i+1} \mid \boldsymbol{c}, x^i) \leq \mathcal{M}_s(x_{i+1} \mid \boldsymbol{c}, x^i)$, then we have in the recursive formula of $p_i$'s, we always have

$$p_i(x^i) = p_{i-1}(x^{i-1})\frac{\mathcal{M}_b(x_i \mid \boldsymbol{c}, x^{i-1})}{\mathcal{M}_s(x_i \mid \boldsymbol{c}, x^{i-1})},$$

and hence

$$p_{\ell-1}(x^{\ell-1}) = \frac{\mathcal{M}_b(x^{\ell-1} \mid \boldsymbol{c})}{\mathcal{M}_s(x^{\ell-1} \mid \boldsymbol{c})},$$

And for $x^\ell$, we have

$$p_\ell(x^\ell) = \min\{\frac{\mathcal{M}_b(x^{\ell} \mid \boldsymbol{c})}{\mathcal{M}_s(x^{\ell} \mid \boldsymbol{c})}, 1\}.$$

This would imply that

$$\begin{aligned}\Pr\left(O^\ell = x^\ell, \tau \geq \ell\right) &= \Pr\left(X^\ell = x^\ell\right)\Pr\left(\tau \geq \ell \mid X^\ell = x^\ell\right) \\ &> \mathcal{M}_s(x^\ell \mid \boldsymbol{c}) \cdot \min\{\frac{\mathcal{M}_b(x^{\ell} \mid \boldsymbol{c})}{\mathcal{M}_s(x^{\ell} \mid \boldsymbol{c})}, 1\} \\ &= \min\{\mathcal{M}_b(x^\ell \mid \boldsymbol{c}), \mathcal{M}_s(x^\ell \mid \boldsymbol{c})\},\end{aligned}$$

which leads to a contradiction.

In the other case, if there exists $i < \ell$ such that, $p_i(x^i) = 1$. Let $i$ be the largest such index. In this case, we have $\mathcal{M}_b(x^i \mid \boldsymbol{c}) > \mathcal{M}_s(x^i \mid \boldsymbol{c})$ since otherwise, we won't have $p_i(x^i) = 1$. And we have

$$p_\ell(x^\ell) = \frac{\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)}{\mathcal{M}_s^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)}.$$

Then by the induction hypothesis, we have

$$\Pr\left(O^i = x^i, \tau \geq i\right) = \Pr\left(X^i = x^i\right)\Pr\left(\tau \geq i \mid X^i = x^i\right) \leq \mathcal{M}_s(x^i \mid \boldsymbol{c}) < \mathcal{M}_b(x^i \mid \boldsymbol{c}).$$

Hence

$$\Pr\left(O^i = x^i, \tau < i\right) = \Pr\left(O^i = x^i\right) - \Pr\left(O^i = x^i, \tau \geq i\right) = \mathcal{M}_b(x^i \mid \boldsymbol{c}) - \mathcal{M}_s(x^i \mid \boldsymbol{c}) > 0.$$

Note that when $O^i = x^i, \tau < i$, by constraints in Equation (6), we have

$$\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau < i\right) = \mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i).$$

This implies

$$\begin{aligned}\Pr\left(O^\ell = x^\ell\right) = \Pr\left(O^i = x^i, \tau < i\right) &\cdot \mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i) \\ &+ \Pr\left(O^i = x^i, \tau \geq i\right)\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau \geq i\right)\end{aligned}$$

Moreover, we have

$$\Pr\left(O^\ell = x^\ell\right) = \mathcal{M}_b(x^i)\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)$$

Combining both, we get

$$\begin{aligned}
1 &= \frac{\Pr\left(O^\ell = x^\ell\right)}{\mathcal{M}_b(x^i)\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)} \\
&= \Pr\left(\tau < i \mid O^i = x^i\right) + \Pr\left(\tau \geq i \mid O^i = x^i\right)\frac{\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau \geq i\right)}{\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)}, \\
&= 1 - \Pr\left(\tau \geq i \mid O^i = x^i\right)\left(\frac{\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau \geq i\right)}{\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)} - 1\right),
\end{aligned}$$

and this implies that

$$\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau \geq i\right) = \mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i). \tag{21}$$

Hence

$$\begin{aligned}
\Pr\left(O^\ell = x^\ell, \tau \geq \ell\right) &\leq \Pr\left(O^\ell = x^\ell, \tau \geq i\right) \\
&= \Pr\left(O^i = x^i, \tau \geq i\right)\Pr\left(O^{i+1:\ell} = x^{i+1:\ell} \mid O^i = x^i, \tau \geq i\right) \\
&\leq \mathcal{M}_s(x^i \mid \boldsymbol{c})p_i(x^i)\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i) \\
&= \mathcal{M}_s(x^i \mid \boldsymbol{c})\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i).
\end{aligned}$$

However, by assumption,

$$\begin{aligned}
\Pr\left(O^\ell = x^\ell, \tau \geq \ell\right) &= \Pr\left(X^\ell = x^\ell\right)\Pr\left(\tau \geq \ell \mid X^\ell = x^\ell\right) \\
&> \mathcal{M}_s(x^\ell \mid \boldsymbol{c})p_\ell(x^\ell) \\
&= \mathcal{M}_s(x^\ell \mid \boldsymbol{c})\frac{\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)}{\mathcal{M}_s^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i)} \\
&= \mathcal{M}_s(x^i \mid \boldsymbol{c})\mathcal{M}_b^{\ell-i}(x^{i+1:\ell} \mid \boldsymbol{c}, x^i),
\end{aligned}$$

which leads to a contradiction. This completes the proof.

## E. Additional Results

In this section, we present experimental results for the same set of experiments described in Section 5 different block lengths ($\gamma = 4, 6, 8$) and different drafters (PALM-2-XXS and PALM-2-XXXS), listed below.

- Table 3. Drafter: PALM-2-XXS, $\gamma = 4$.
- Table 4. Drafter: PALM-2-XXS, $\gamma = 6$.
- Table 5. Drafter: PALM-2-XXXS, $\gamma = 4$.
- Table 6. Drafter: PALM-2-XXXS, $\gamma = 6$.
- Table 7. Drafter: PALM-2-XXXS, $\gamma = 8$.

*Table 3.* Speedup comparison between token verification (TokenV) and block verification (BlockV) with $\gamma = 4$ and PALM-2-XXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TokenV | BlockV | Improve. ↑ % | TokenV | BlockV | Improve. ↑ % |
| LM1B | $2.78 \pm 0.01$ | $2.88 \pm 0.01$ | $3.48 \pm 0.24$ | $2.36 \pm 0.00$ | $2.42 \pm 0.01$ | $2.51 \pm 0.22$ |
| GPT Prompt | $2.88 \pm 0.01$ | $3.00 \pm 0.00$ | $4.33 \pm 0.25$ | $2.43 \pm 0.01$ | $2.51 \pm 0.00$ | $3.43 \pm 0.24$ |
| WebQA | $2.91 \pm 0.01$ | $2.99 \pm 0.01$ | $2.83 \pm 0.65$ | $2.45 \pm 0.01$ | $2.50 \pm 0.01$ | $1.94 \pm 0.61$ |
| PIQA | $2.89 \pm 0.00$ | $2.99 \pm 0.01$ | $3.48 \pm 0.21$ | $2.44 \pm 0.00$ | $2.50 \pm 0.01$ | $2.66 \pm 0.20$ |
| ShareGPT | $2.85 \pm 0.01$ | $2.95 \pm 0.00$ | $3.48 \pm 0.19$ | $2.41 \pm 0.01$ | $2.47 \pm 0.00$ | $2.63 \pm 0.17$ |
| XSum | $2.94 \pm 0.01$ | $3.03 \pm 0.01$ | $3.24 \pm 0.51$ | $2.48 \pm 0.01$ | $2.54 \pm 0.01$ | $2.35 \pm 0.48$ |
| GSM8K | $3.12 \pm 0.01$ | $3.21 \pm 0.02$ | $3.06 \pm 0.95$ | $2.62 \pm 0.01$ | $2.68 \pm 0.02$ | $2.19 \pm 0.89$ |
| WMT-DeEn | $2.75 \pm 0.01$ | $2.83 \pm 0.01$ | $2.99 \pm 0.09$ | $2.33 \pm 0.01$ | $2.38 \pm 0.01$ | $2.18 \pm 0.09$ |
| Average | 2.89 | 2.99 | 3.36 | 2.44 | 2.50 | 2.49 |

*Table 4.* Speedup comparison between token verification (TokenV) and block verification (BlockV) with $\gamma = 6$ and PALM-2-XXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TokenV | BlockV | Improve. ↑ % | TokenV | BlockV | Improve. ↑ % |
| LM1B | $3.08 \pm 0.01$ | $3.27 \pm 0.01$ | $6.42 \pm 0.07$ | $2.32 \pm 0.01$ | $2.43 \pm 0.01$ | $5.00 \pm 0.06$ |
| GPT Prompt | $3.22 \pm 0.01$ | $3.44 \pm 0.02$ | $6.55 \pm 0.83$ | $2.42 \pm 0.00$ | $2.54 \pm 0.02$ | $5.06 \pm 0.77$ |
| WebQA | $3.26 \pm 0.01$ | $3.44 \pm 0.01$ | $5.60 \pm 0.22$ | $2.45 \pm 0.01$ | $2.55 \pm 0.01$ | $4.24 \pm 0.21$ |
| PIQA | $3.22 \pm 0.02$ | $3.43 \pm 0.02$ | $6.36 \pm 0.78$ | $2.42 \pm 0.01$ | $2.54 \pm 0.01$ | $4.92 \pm 0.72$ |
| ShareGPT | $3.18 \pm 0.02$ | $3.37 \pm 0.01$ | $6.13 \pm 0.53$ | $2.39 \pm 0.02$ | $2.50 \pm 0.01$ | $4.74 \pm 0.49$ |
| XSum | $3.29 \pm 0.01$ | $3.48 \pm 0.01$ | $5.91 \pm 0.82$ | $2.47 \pm 0.01$ | $2.58 \pm 0.01$ | $4.47 \pm 0.77$ |
| GSM8K | $3.56 \pm 0.01$ | $3.80 \pm 0.03$ | $6.86 \pm 0.60$ | $2.66 \pm 0.01$ | $2.80 \pm 0.02$ | $5.38 \pm 0.56$ |
| WMT-DeEn | $3.04 \pm 0.01$ | $3.19 \pm 0.01$ | $4.92 \pm 0.29$ | $2.29 \pm 0.01$ | $2.37 \pm 0.01$ | $3.57 \pm 0.27$ |
| Average | 3.23 | 3.43 | 6.10 | 2.43 | 2.54 | 4.67 |

*Table 5.* Speedup comparison between token verification (TokenV) and block verification (BlockV) with $\gamma = 4$ and PALM-2-XXXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TokenV | BlockV | Improve. ↑ % | TokenV | BlockV | Improve. ↑ % |
| LM1B | $2.24 \pm 0.00$ | $2.33 \pm 0.01$ | $4.23 \pm 0.44$ | $2.25 \pm 0.00$ | $2.34 \pm 0.01$ | $3.89 \pm 0.41$ |
| GPT Prompt | $2.41 \pm 0.02$ | $2.48 \pm 0.01$ | $2.96 \pm 1.00$ | $2.42 \pm 0.02$ | $2.48 \pm 0.01$ | $2.72 \pm 0.94$ |
| WebQA | $2.38 \pm 0.01$ | $2.45 \pm 0.01$ | $2.87 \pm 0.13$ | $2.39 \pm 0.01$ | $2.45 \pm 0.01$ | $2.63 \pm 0.12$ |
| PIQA | $2.36 \pm 0.01$ | $2.43 \pm 0.01$ | $3.22 \pm 0.37$ | $2.37 \pm 0.01$ | $2.44 \pm 0.01$ | $2.97 \pm 0.35$ |
| ShareGPT | $2.34 \pm 0.00$ | $2.42 \pm 0.01$ | $3.49 \pm 0.12$ | $2.35 \pm 0.00$ | $2.42 \pm 0.01$ | $3.16 \pm 0.12$ |
| XSum | $2.38 \pm 0.01$ | $2.45 \pm 0.01$ | $2.91 \pm 0.63$ | $2.39 \pm 0.01$ | $2.45 \pm 0.01$ | $2.68 \pm 0.60$ |
| GSM8K | $2.51 \pm 0.01$ | $2.58 \pm 0.02$ | $2.99 \pm 0.47$ | $2.51 \pm 0.01$ | $2.58 \pm 0.02$ | $2.74 \pm 0.44$ |
| WMT-DeEn | $2.22 \pm 0.00$ | $2.28 \pm 0.00$ | $2.59 \pm 0.09$ | $2.24 \pm 0.00$ | $2.29 \pm 0.00$ | $2.37 \pm 0.08$ |
| Average | 2.35 | 2.43 | 3.16 | 2.36 | 2.43 | 2.89 |

*Table 6.* Speedup comparison between token verification (TOKENV) and block verification (BLOCKV) with $\gamma = 6$ and PALM-2-XXXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TOKENV | BLOCKV | Improve. $\uparrow$ % | TOKENV | BLOCKV | Improve. $\uparrow$ % |
| LM1B | $2.36 \pm 0.01$ | $2.48 \pm 0.00$ | $4.93 \pm 0.46$ | $2.27 \pm 0.01$ | $2.37 \pm 0.00$ | $4.55 \pm 0.43$ |
| GPT Prompt | $2.58 \pm 0.04$ | $2.72 \pm 0.02$ | $5.57 \pm 1.29$ | $2.46 \pm 0.03$ | $2.59 \pm 0.01$ | $5.10 \pm 1.22$ |
| WebQA | $2.54 \pm 0.00$ | $2.68 \pm 0.02$ | $5.46 \pm 0.50$ | $2.43 \pm 0.00$ | $2.55 \pm 0.01$ | $5.02 \pm 0.47$ |
| PIQA | $2.50 \pm 0.00$ | $2.62 \pm 0.01$ | $5.06 \pm 0.39$ | $2.39 \pm 0.00$ | $2.50 \pm 0.01$ | $4.66 \pm 0.37$ |
| ShareGPT | $2.47 \pm 0.01$ | $2.60 \pm 0.01$ | $5.10 \pm 0.49$ | $2.37 \pm 0.01$ | $2.48 \pm 0.01$ | $4.69 \pm 0.46$ |
| XSum | $2.54 \pm 0.01$ | $2.67 \pm 0.01$ | $4.83 \pm 0.47$ | $2.43 \pm 0.01$ | $2.54 \pm 0.01$ | $4.45 \pm 0.44$ |
| GSM8K | $2.71 \pm 0.03$ | $2.83 \pm 0.00$ | $4.27 \pm 0.89$ | $2.58 \pm 0.02$ | $2.69 \pm 0.00$ | $3.92 \pm 0.84$ |
| WMT-DeEn | $2.31 \pm 0.01$ | $2.43 \pm 0.02$ | $5.38 \pm 0.57$ | $2.21 \pm 0.00$ | $2.32 \pm 0.01$ | $4.99 \pm 0.54$ |
| Average | 2.50 | 2.63 | 5.07 | 2.39 | 2.50 | 4.67 |

*Table 7.* Speedup comparison between token verification (TOKENV) and block verification (BLOCKV) with $\gamma = 8$ and PALM-2-XXXS being the draft model. Each statistic is computed using 1000 test prompts from different datasets on various tasks (each run is an average with 3 different random seeds). Numbers after $\pm$ represent standard deviation.

| Dataset | Block efficiency | | | Wall clock time speedup over baseline | | |
|---|---|---|---|---|---|---|
| | TOKENV | BLOCKV | Improve. $\uparrow$ % | TOKENV | BLOCKV | Improve. $\uparrow$ % |
| LM1B | $2.40 \pm 0.01$ | $2.55 \pm 0.01$ | $6.19 \pm 0.43$ | $2.13 \pm 0.01$ | $2.25 \pm 0.01$ | $5.28 \pm 0.40$ |
| GPT Prompt | $2.66 \pm 0.01$ | $2.82 \pm 0.02$ | $6.28 \pm 1.01$ | $2.35 \pm 0.01$ | $2.47 \pm 0.02$ | $5.37 \pm 0.95$ |
| WebQA | $2.61 \pm 0.01$ | $2.78 \pm 0.00$ | $6.27 \pm 0.49$ | $2.31 \pm 0.01$ | $2.43 \pm 0.00$ | $5.39 \pm 0.46$ |
| PIQA | $2.57 \pm 0.01$ | $2.76 \pm 0.01$ | $7.48 \pm 0.51$ | $2.27 \pm 0.01$ | $2.42 \pm 0.01$ | $6.51 \pm 0.47$ |
| ShareGPT | $2.54 \pm 0.01$ | $2.71 \pm 0.01$ | $6.63 \pm 0.72$ | $2.25 \pm 0.01$ | $2.38 \pm 0.01$ | $5.68 \pm 0.68$ |
| XSum | $2.60 \pm 0.01$ | $2.77 \pm 0.00$ | $6.46 \pm 0.49$ | $2.30 \pm 0.01$ | $2.43 \pm 0.00$ | $5.53 \pm 0.46$ |
| GSM8K | $2.82 \pm 0.02$ | $2.98 \pm 0.03$ | $5.48 \pm 1.18$ | $2.49 \pm 0.01$ | $2.60 \pm 0.03$ | $4.62 \pm 1.11$ |
| WMT-DeEn | $2.37 \pm 0.00$ | $2.49 \pm 0.01$ | $5.33 \pm 0.46$ | $2.10 \pm 0.00$ | $2.20 \pm 0.01$ | $4.53 \pm 0.43$ |
| Average | 2.57 | 2.73 | 6.27 | 2.28 | 2.40 | 5.36 |