An Exploration of Hierarchical Speculative Decoding

Haotian Chen

Abstract

In this paper, we propose a variant of *speculative decoding* that has a *hierarchical* structure. We design a relatively simple extension from the original version and analyze the theoretical improvements. Unfortunately, while the new method shows better performance under certain configuration, it is less efficient in most practical cases. The main reason is that the *reject step* in the algorithm makes the increment of samples from smaller models with lower *acceptance rate* meaningless. Although it might be an unsuccessful attempt, the process of exploration may bring inspiration and deepen the understanding of the collaborative effects between large language models.

1 Introduction

While large language models have achieved extraordinary performance in numerous domains, their huge size brings greater cost and slower speed than small models. To improve the efficiency, many different approaches have been developed to accelerate the inference.

Speculative decoding, compared to other methods, is a powerful tool to sample from autoregressive models faster without any changes to the outputs, and can also be easily applied to various models. It combines advantages of a powerful target model and an efficient approximation model and shows good speedup in multiple tasks[1][2].

The idea of considering cooperation between language models has inspired us. Since two different sizes of models can bring significant improvements, is it possible to go even further by adding a smaller level to form a system of hierarchy? This is the basic problem we try to answer in our paper.

In summary, our main results are as follows:

- We propose a hierarchical variant of speculative decoding that has not been studied;
- Though unable to conduct experiments, we analyze the theoretical performance of our algorithm and compare it with the original version.

2 Method

We now introduce our method—a hierarchical version of speculative decoding for accelerating language models. To begin with, we will present the original algorithm and introduce some notations.

2.1 Speculative Decoding

Let M_p be the target large model that we try to accelerate, M_q be a more efficient approximation model for the same task, and denote by $p(x_t|x_{<t}), q(x_t|x_{<t})$ the distribution we get from the models for a prefix $x_{<t}$ respectively[1].

[1]We'll use $p(x)$ to denote $p(x_t|x_{<t})$ whenever the prefix $x_{<t}$ is clear from the context, and similarly for $q(x)$.

Preprint. Under review.

At a high level, the core ideas of speculative decoding are as follows:

1. **Sample from M_q.** Use the more efficient model M_q to generate $\gamma \in \mathbb{Z}^+$ guesses.
2. **Evaluate with M_p.** Use the target model M_p to evaluate all of the guesses and their respective probabilities from M_q *in parallel*, accepting all those that *can* lead to an identical distribution.
3. **Adjust distribution.** Sample an additional token from an adjusted distribution to fix the first one that was rejected, or to add an additional one if they are all accepted

Algorithm 1 Speculative Decoding

Inputs: $M_p, M_q, prefix$.
for $i = 1$ **to** γ **do** ▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$ ▷ Run M_p in parallel.
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$ ▷ Determine the number of accepted guesses n.
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
$p'(x) \leftarrow p_{n+1}(x)$ ▷ Adjust the distribution from M_p if needed.
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
$t \sim p'(x)$ ▷ Return one token from M_p, and n tokens from M_q.
return $prefix + [x_1, \ldots, x_n, t]$

Algorithm 1 is effective as it generates more tokens at once, and since M_q are more efficient, the total time will not increase significantly. So, the average time required to generate each token reduces.

2.2 Hierarchical Variant

To form a hierarchical structure, we add a smaller model M_s. And for every guess generated by the medium model M_q, M_s will sample n guesses autoregressively. Similarly, each time the large model M_p produces one token, there will be m samples from M_q. Then, the *reject step* will determine the number of accepted samples.

Algorithm 2 Hierarchical Speculative Decoding

Inputs: $M_p, M_q, M_s, prefix$.
for $i = 1$ **to** m **do** ▷ Sample m guesses from M_q and mn guesses from M_s autoregressively.
  for $j = 1$ **to** n **do**
    $s_{i,j}(x) \leftarrow M_s(prefix + [x_{1,1}, \ldots, x_{i,j-1}])$
    $x_{i,j} \sim s_{i,j}(x)$
  $q_i(x) \leftarrow M_q(prefix + [x_{1,1}, \ldots, x_{i,j}])$
  $x_{i,n+1} \sim q_i(x)$
$p_{1,1}(x), \ldots, p_{m,n+1}(x), p_{m(n+1)+1}(x) \leftarrow$ ▷ Run M_p in parallel.
  $M_p(prefix), \ldots, M_p(prefix + [x_{1,1}, \ldots, x_{m,n+1}])$
$r_{1,1} \sim U(0,1), \ldots, r_{m,n+1} \sim U(0,1)$ ▷ Determine the number of accepted guesses k.
$k_s \leftarrow \min(\{(i-1)(n+1) + j \mid 1 \le i \le m, 1 \le j \le n, r_{i,j} > \frac{p_{i,j}(x)}{s_{i,j}(x)}\} \cup \{m(n+1)+1\})$
$k_q \leftarrow \min(\{(i-1)(n+1) + j \mid 1 \le i \le m, j = n+1, r_{i,j} > \frac{p_{i,j}(x)}{q_i(x)}\})$
$k \leftarrow \min(k_s, k_q)$
$p'(x) \leftarrow p_{k+1}(x)$
$t \sim p'(x)$ ▷ Return one token from M_p, and k tokens from M_q or M_s.
return $prefix + [x_{1,1}, \ldots, x_k, t]$

Here, for the convenience of later analysis, we just put the *reject step* after generating all guesses from M_q and M_s. And we have to admit that the distribution of outputs may differ from M_p (not worse).

3 Analysis

3.1 Number of Generated Tokens

In this part, we will analyze the expected number of tokens produced by a single run of Algorithm 2. And we will introduce definitions and results of standard speculative decoding at first.

Definition 3.1. The *acceptance rate*

$\rho_{x_{<t}}$ is the probability of accepting $x_t \sim q(x_t|x_{<t})$ by Algorithm 1 (given a prefix $x_{<t}$)[2].

$\alpha \stackrel{\text{def}}{=} E(\rho)$ measures how well M_q approximates M_p.

Then, if we apply the simplifying assumptions that ρs are i.i.d., we will get the expected number of tokens generated by Algorithm 1 as follows:

$$E(\text{generated tokens}) = \frac{1-\alpha^{\gamma+1}}{1-\alpha} \tag{1}$$

For the hierarchical version, we denote β as the expected probability of accepting a sample from the small model M_s, while α remains the same. And we can get the following result:

Theorem 3.2. Expected tokens generated from Algorithm 2

$$E(\text{generated tokens}) = \frac{1-\beta^{n+1}}{1-\beta} \cdot \frac{1-Q^m}{1-Q} + Q^m, Q = \alpha \cdot \beta^n \tag{2}$$

Proof of Theorem 3.2 requires some tedious calculations, so the details are attached in the appendix. Here, we can consider several special circumstances to get an intuitive understanding:

- If $n = 0$, the result comes to be $E = \frac{1-Q^{m+1}}{1-Q}$ and $Q = \alpha$, which is equivalent to Equation (1), that only use the medium model M_q to accelerate.
- If $m = 1$ and let $\alpha = \beta$, we will get $E = \frac{1-\beta^{n+2}}{1-\beta}$, which is equal to only use the small model M_s to sample $n + 1$ guesses faster.
- If $n \to \infty$, then $E \to \frac{1}{1-\beta}$, which means the limit (*not the same as the best*) of the performance of Algorithm 2 depends on the small model M_s.

3.2 Time Improvement

We have shown that with i.i.d. assumption, hierarchical speculative decoding will contribute to accelerating as Theorem 3.2. Now let us consider the total time improvement. Since speculative execution assumes that we have enough compute resources, we will omit the time increased by evaluations of M_p.

Definition 3.3. Let c be the cost coefficient—the ratio between the time for a single run of M_q and M_p, and c_s, similarly, denote the ratio of time between M_s and M_q.

Theorem 3.4. The expected improvement factor in total time by Algorithm 2 is

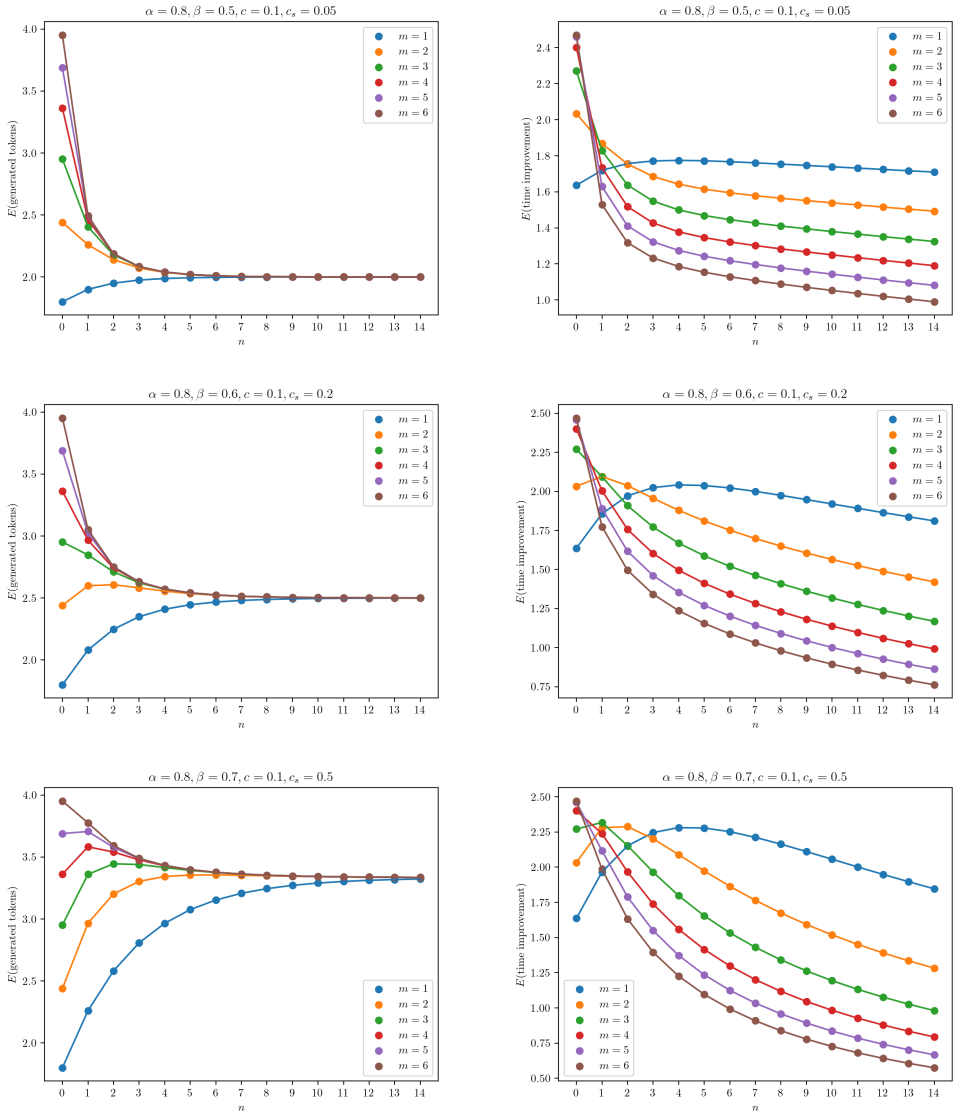
$$E(\text{time improvement}) = \frac{\frac{1-\beta^{n+1}}{1-\beta} \cdot \frac{1-Q^m}{1-Q} + Q^m}{(1 + mc + mncc_s)}, Q = \alpha \cdot \beta^n \tag{3}$$

Proof. Denote T as the time required for a single step of M_p. Then, each run of Algorithm 2 costs $T + mcT + mncc_sT$ (for sampling from the small model M_s mn times, from medium model M_q m times and from large model M_p once). Combine it with Equation (2), we can easily obtain the result. □

[2]As before, we'll omit the $x_{<t}$ subscript whenever the prefix is clear from the context.

4 Results

Our main results are as follows. Due to a lack of hardware support, we only further analyze the performance of Algorithm 2 in some specific scenarios and compare it with the original algorithm.



(a) $E(\text{generated tokens})$ with different β and c_s (b) $E(\text{time improvement})$ with different β and c_s

Figure 1: Theoretical performance of Algorithm 2

Here, we include $n = 0$ as our baseline, which is equivalent to only use M_q to produce m guesses. Since we mainly care about the comparison between Algorithm 2 and Algorithm 1, we just choose a fixed but representative α and consider the effects of the changes from the small model M_s.

And we can conclude from Figure 1 that:

- Under most practical cases, introducing a smaller model M_s will not lead to a better performance, and as n—the number of samples from M_s increases, the expected time improvement tends to decrease in general, some even reduce to less than 1. Similar for the expected generated tokens.
- While as β (along with c_s) increases, which means the better small model M_s we choose, will permit more choices of m that can bring a further speed up, compared with only using M_q to accelerate.

Before analyzing the possible causes of the results, we can further understand our method through the following example. It is an idealized but vivid situation that shows how the acceleration happens.

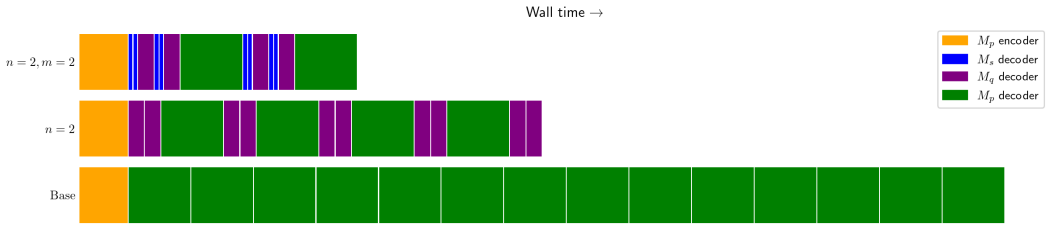


Figure 2: A simplified trace diagram to understand speculative decoding.

Now, let us consider the reason that why we do not observe a significant improvement by Algorithm 2.

The direct reason is that, according to our computation and analysis of Theorem 3.2, the expected generated tokens has a limitation depends on β, which is of course less than α. However, Algorithm 2 brings more operations, which leads to the ultimate improvement less efficient.

To further explain it, we can regard speculative decoding as a kind of arrangement to put in M_s or M_q decoders to the long M_p sequence. If it works well, it should bring speedup. But, if you just add too many layers that are less possible to work, here M_s, at the very beginning, then since the process is autoregressive, later samples from M_q are less likely to be accepted, which results in worse performance of Algorithm 2.

5 Conclusion

In this work, we explore a new hierarchical version of speculative decoding for accelerating language models. We try to combine three levels of models, large, medium and small, to achieve faster speedup. Due to hardware limitations, we are not able to conduct experimental verification and only presented theoretical results. And the analysis is based on the i.i.d. assumption of accepted tokens, so the results may not match real experiments. Another flaw is that we have not found a perfect method to ensure the distribution of outputs exactly the same as the large model alone.

According to our analysis and results, under the framework proposed in this paper, adding a smaller model to form a hierarchical structure does not bring noticeable improvement to the acceleration in most cases. It seems to be unexpected, but plausible, because as the number of samples from small model increases, the arithmetic operations rise along with the probability of accepting all these samples decreases significantly, which means most of the extra samples are meaningless and will not enhance efficiency.

To address the problem, one possible solution is to choose a less rigorous reject rule (especially for the small model), which is likely to sacrifice the quality of the final output (depends on the adjust step). And there may be better hierarchical framework designs left to be explored.

References

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

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

A Proof of Theorem 3.2

Proof. Let X denotes the number of tokens generated after a single run of Algorithm 2, then X ranges from 1 to $m(n+1)+1$ and we can get its probability distribution as follows:

Table A.1: Probability distribution of X

X	1	2	$\cdots$	n	$n+1$
P	$1-\beta$	$\beta(1-\beta)$	$\cdots$	$\beta^{n-1}(1-\beta)$	$\beta^n(1-\alpha)$
X	$n+2$	$n+3$	$\cdots$	$2n+1$	$2n+2$
P	$\beta^n(1-\beta)\alpha$	$\beta^{n+1}(1-\beta)\alpha$	$\cdots$	$\beta^{2n-1}(1-\beta)\alpha$	$\beta^{2n}\alpha(1-\alpha)$
			$\cdots$		
X	$(m-1)(n+1)+1$	$(m-1)(n+1)+2$	$\cdots$	$m(n+1)-1$	$m(n+1)$
P	$\beta^{n(m-1)}(1-\beta)\alpha^{m-1}$	$\beta^{n(m-1)+1}(1-\beta)\alpha^{m-1}$	$\cdots$	$\beta^{mn-1}(1-\beta)\alpha^{m-1}$	$\beta^{mn}\alpha^{m-1}(1-\alpha)$
X	$m(n+1)+1$				
P	$\beta^{mn}\alpha^m$				

So we just need to calculate $E(X)$.

For the convenience of subsequent calculations, we introduce the following notations:

$$P_i \overset{\text{def}}{=} \sum_{k=(i-1)(n+1)+1}^{i(n+1)} P(X=k), 1 \le i \le m \tag{4}$$

$$S_i \overset{\text{def}}{=} \sum_{k=(i-1)(n+1)+1}^{i(n+1)} k \cdot P(X=k), 1 \le i \le m \tag{5}$$

Then

$$E(X) = S_1 + S_2 + \cdots + S_m + [m(n+1)+1] \cdot \beta^{mn} \cdot \alpha^m \tag{6}$$

Notice the recurrence relation of the probability distribution in Table A.1, we have

$$P(X = k+n+1) = P(X=k) \cdot \beta^n\alpha \Longrightarrow P_{i+1} = P_i \cdot \beta^n\alpha = P_i \cdot Q \tag{7}$$

Note that

$$\begin{aligned} S_{i+1} &= \sum_{k=i(n+1)+1}^{(i+1)(n+1)} k \cdot P(X=k) \\ &= \sum_{k=i(n+1)+1}^{(i+1)(n+1)} [k-(n+1)+(n+1)] \cdot P(X=k) \\ &= \sum_{k-(n+1)=(i-1)(n+1)+1}^{i(n+1)} [k-(n+1)] \cdot P(X=k-(n+1)) \cdot Q + (n+1)P_{i+1} \\ &= S_i \cdot Q + (n+1)P_{i+1} \end{aligned} \tag{8}$$

After calculating S_1 and P_1, we will obtain $E(X)$ by the recurrence relation of (8) and (7).

For P_1, observe that $\{P_i\}$ is a geometric sequence, and $Q = \beta^n\alpha$ is the common ratio, we can get

$$1 = \sum_{k=1}^{m(n+1)+1} P(X=k) = \sum_{i=1}^{m} P_i + \beta^{mn}\alpha^m = \frac{P_1(1-Q^m)}{1-Q} + Q^m \Longrightarrow P_1 = 1-Q \tag{9}$$

And for S_1, it is easy to get that

$$\begin{aligned} S_1 =&(1-\beta)+2\beta(1-\beta)+\cdots+n\beta^{n-1}(1-\beta)+(n+1)\beta^n(1-\alpha) \\ =&(1-\beta)[1+2\beta+\cdots+n\beta^{n-1}]+(n+1)\beta^n(1-\alpha) \\ =&\frac{1-\beta^n}{1-\beta}-n\beta^n+(n+1)\beta^n-(n+1)\beta^n\alpha \\ =&\frac{1-\beta^{n+1}}{1-\beta}-(n+1)\beta^n\alpha \end{aligned} \tag{10}$$

Divide both sides of Equation (8) by Q^{i+1}, we get

$$\frac{S_{i+1}}{Q^{i+1}}=\frac{S_i}{Q^i}+(n+1)\frac{P_1}{Q} \tag{11}$$

So, $\left\{\frac{S_i}{Q^i}\right\}$ is an arithmetic sequence, hence

$$S_{i+1}=S_1\cdot Q^i+iQ^iP_1(n+1) \tag{12}$$

Finally, apply (9), (10) and (12) to (6), we get the result

$$\begin{aligned} E(x) =&S_1+S_1Q+\cdots S_1Q^{m-1}+(n+1)P_1\sum_{i=1}^{m-1}iQ^i+[m(n+1)+1]\cdot\beta^{mn}\cdot\alpha^m \\ =&S_1\cdot\frac{1-Q^m}{1-Q}+(n+1)Q\left[\frac{1-Q^{m-1}}{1-Q}-(m-1)Q^{m-1}\right]+[m(n+1)+1]\cdot Q^m \\ =&S_1\cdot\frac{1-Q^m}{1-Q}+(n+1)Q\cdot\frac{1-Q^{m-1}}{1-Q}+(n+2)Q^m \\ =&S_1\cdot\frac{1-Q^m}{1-Q}+(n+1)\left[\frac{Q-Q^m}{1-Q}+Q^m\right]+Q^m \\ =&\left[\frac{1-\beta^{n+1}}{1-\beta}-(n+1)\beta^n\alpha\right]\cdot\frac{1-Q^m}{1-Q}+(n+1)Q\cdot\frac{1-Q^m}{1-Q}+Q^m \\ =&\frac{1-\beta^{n+1}}{1-\beta}\cdot\frac{1-Q^m}{1-Q}+Q^m \end{aligned} \tag{13}$$

□