Higher Acceptance Rates for Speculative Decoding with Randomised Drafting

William Toner [1] **Martin Asenov** [1] **Rajkarn Singh** [1] **Artjom Joosen** [1]

Abstract

Speculative decoding is an approach for increasing the Tokens Per Second (TPS) of a base LLM by using a smaller draft model to predict subsequent tokens. These draft tokens can be generated quickly and their verification by the base model can occur in parallel with generating the next token. A key determinant of the impact of SD on TPS is the *acceptance rate* — the probability that a draft token will be accepted upon verification.

This work explores **Randomised Drafting** wherein a draft is only generated with some probability $a \leq 1$. By introducing this random component, we show that the acceptance rate can be boosted while preserving the distributional guarantees of SD. Despite sometimes using the base model directly, we show that Randomised Drafting can result in an overall boost in TPS. The improvement in TPS is minor but comes without cost.

1. Introduction

Large language models (LLMs) have become a key technology over the past decade, with growing adoption across many applications. As these models scale, so do their computational requirements. A major bottleneck is their autoregressive nature: tokens must be generated one at a time, which can make inference slow. This limits how quickly LLMs can produce output—measured in tokens per second (TPS)—and affects user experience. Faster generation is increasingly important as demand grows, motivating efforts to improve the efficiency of LLM decoding.

Speculative Decoding (SD) has been developed to address the requirement for higher TPS. A significant property of SD is that one can ensure no alteration in the output distribution relative to using the base model by itself. This makes SD more attractive than approaches like model distillation which alter the output distribution and may perform worse on certain tasks. SD works by introducing a smaller draft model that proposes several tokens at once, which are then either accepted or rejected by the base model - a process known as verification. Verification happens in parallel with generation, so when most draft tokens are accepted, TPS approaches that of the faster draft model. Conversely, if too many drafts are rejected, SD can underperform, even slowing down inference. As a result, the effectiveness of SD requires optimising the acceptance rate of the draft tokens.

1.1. Contributions

This work introduces Speculative Decoding with Randomised Drafting (RD). We demonstrate theoretically and experimentally how RD allows higher acceptance rates than vanilla SD while satisfying the fidelity property of vanilla SD. We theoretically analyse the impact on TPS of RD showing that may yield a small gain.

2. Related Work

Speculative decoding is a key technique for accelerating inference in large language models (LLMs) without sacrificing output quality (Leviathan et al., 2023). It offers statistical guarantees on output quality, building on earlier concepts of block decoding and verification (Stern et al., 2018) which lacked these guarantees. The fundamental idea involves using a smaller, faster 'draft' model to generate a sequence of candidate tokens. These tokens are then verified in parallel by the larger, more powerful 'target' model (Fu et al., 2024). This method aims to reduce the sequential dependency inherent in autoregressive generation, thereby decreasing latency and improving throughput. Importantly, the generated outputs are distributed identically to those drawn directly from the target model.

Significant research efforts have focused on improving the efficiency of speculative decoding (Hu et al., 2025). A primary area of investigation is the optimization of the draft proposal mechanism. This includes strategies for determining the optimal number of draft tokens, k, to generate in each step (Gloeckle et al., 2024). Beyond fixed lengths, dynamic adaptation of the candidate chain length based on contextual cues or model confidence has also been explored

[1]Huawei SIR Lab, Edinburgh Research Centre, UK. Correspondence to: William Toner <william.toner2@huawei.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

to further boost performance (Huang et al., 2024; Liu et al., 2024b). Another approach involves generating multiple draft sequences or utilizing multiple specialized decoding heads from the draft model. Examples include methods like Medusa and Hydra (Cai et al., 2024; Ankner et al., 2024). These techniques aim to increase the probability of accepting longer sequences of tokens per verification step.

More advanced strategies include self-decoding, where early layers of the target LLM itself generate drafts. This potentially reduces the need for a separate draft model reducing resource utilization, as seen in systems like DeepSeek-V3 (Liu et al., 2024a; Li et al., 2024). Tree-based speculative decoding methods build a tree of possible continuations from one or more models, allowing for exploration of more diverse candidate sequences (Miao et al., 2024; Wang et al., 2025). Researchers are also actively combining speculative decoding with blockwise parallel decoding principles. The goal here is to break sequential dependencies even further and achieve greater parallelism during the generation process (Kim et al., 2024; Yang et al., 2023; Monea et al., 2023; Narasimhan et al., 2024).

3. Background

3.1. Notation

We list the notation used in this paper below. For the sake of simplicity we will discuss the case where the draft model generates a draft of length $k = 1$.

- z: Prompt tokens
- x_n: The n^{th} reponse token output by the model
- $p(x \mid z), q(x \mid z)$: Base and draft models respectively, assumed to be LLMs
- $\alpha(z)$ Acceptance rate; the probability that a draft token will be accepted given prompt z
- a: Draft probability
- $p_{res}(x \mid z)$ the residual distribution
- k: Draft chain length (assumed to be 1 for simplicity)
- V: Vocab size - total number of tokens

We occasionally abuse notation for brevity. For instance, we sometimes omit dependencies, writing $p(x)$ in place of $p(x \mid z)$. In other cases, we compress notation further by letting $\boldsymbol{p}$ and $\boldsymbol{q}$ denote the probability vectors representing the predicted token distributions of the base and draft models, respectively.

3.2. Vanilla SD

As previously discussed Speculative Decoding works by having a draft model q generate draft tokens which are then verified by the base model p. Specifically, we sample a draft $\tilde{x}_n \sim q(x)$. We then compute $p(\tilde{x}_n \mid z)$ and accept $\tilde{x}_n$ with probability $\alpha = \min\left(1, \frac{p(\tilde{x}|z)}{q(\tilde{x}|z)}\right)$ otherwise rejecting this sample and obtaining a replacement from the *residual* distribution defined

$$p_{res}(x) = norm(max(p(x) - q(x), 0)). \tag{1}$$

This construction ensures that SD satisfies a property we will refer to as **fidelity** where samples obtained through SD and those sampled directly from the base model are indistinguishable. An algorithmic description of SD in given in Algorithm 1.

Fidelity: When the outputs of a Speculative Decoding variant are distributed identically to those of the base model we say that this approach satisfies the Fidelity property.

Algorithm 1 Vanilla SD

1: **Input:** Base model $p(x \mid z)$, draft model $q(x \mid z)$, prompt z
2: Sample draft $\tilde{x} \sim q(x \mid z)$
3: Compute acceptance probability $\alpha = \min\left(1, \frac{p(\tilde{x}|z)}{q(\tilde{x}|z)}\right)$
4: With probability α, set $x \leftarrow \tilde{x}$
5: Otherwise, sample $x \sim p_{\text{res}}(x \mid z)$ {As defined in Eqn. 1}
6: **Output:** Sample x

In order to ensure that SD satsifies fidelity, some tokens drafted by the draft model must be rejected. To illustrate intuitively why this is necessary suppose that for a given prompt z the probability of the next token being `"cat"` is $p(\texttt{"cat"} \mid z) = 0.4$ for the base model and $q(\texttt{"cat"} \mid z) = 0.8$ for the draft model. If we sample from the draft model without discarding at least half of the `"cat"` tokens we will oversample this word relative to the desired frequency. The necessity of discarding a certain proportion of draft tokens to avoid oversampling mitigates the speed-up which might otherwise be realised by SD.

4. Randomised Drafting

Our proposal is that, during inference, we only generate a draft with the draft model with probability $a \in [0, 1]$. I.e. with probability a we draft the next token from $\tilde{x} \sim q(x)$ and with probability $1 - a$ the next token is not drafted. We call this **Randomised Drafting**. By only sometimes sampling from the draft model we mitigate the oversampling problem. To illustrate this let us return to the previous example where $p(\texttt{"cat"} \mid z) = 0.4$ and $q(\texttt{"cat"} \mid z) = 0.8$ and suppose that we only draft with probability 0.5. Even if we accept all `"cat"` tokens output by the draft model we still do not oversample this token. Consequently, random

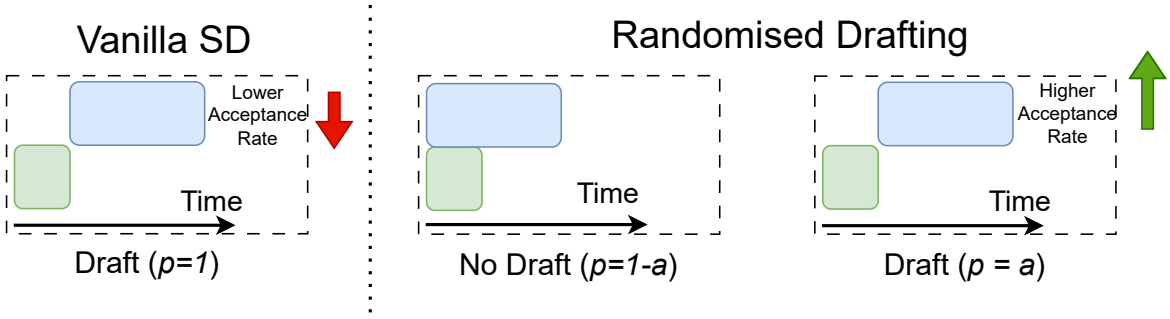

Figure 1. **A comparison of vanilla SD (left) with RD SD (right):** The figure compares vanilla SD with SD employing Randomised Drafting. In vanilla SD the base model 'waits' for the draft model to produce a token and then verifies this token. In the RD setting there are two possibilities: **i)** The *'draft'* setting occurs with probability a and is identical to the vanilla SD setting albeit with a higher acceptance rate. **ii)** In the *'no-draft'* case (occuring with probability $1-a$) the base model and draft model are run in parallel so the base model does not need to wait.

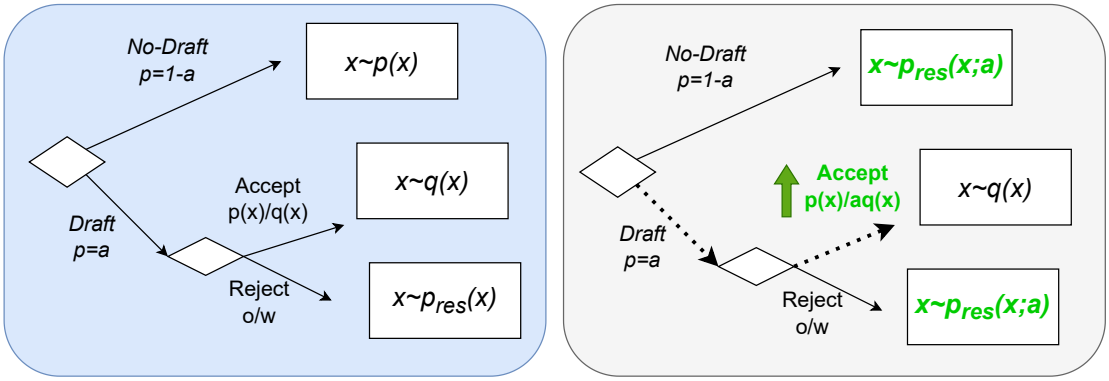

Figure 2. **Two possible ways one could do Randomised Drafting**: A 'naive' approach (left) where we sample directly from the base model $p(x)$ with probability $1-a$ and otherwise apply vanilla SD. Our insight is that this may be improved upon. Our proposed approach (right) shows that we can boost the acceptance rate in this setup while preserving fidelity. The base distribution and residual distributions must be altered ($p(x), p_{res}(x) \mapsto p_{res}(x;a), p_{res}(x;a)$ respectively). Expressions which differ in our variant (relative to the naive approach) are highlighted in green.

drafting allows us to increase the acceptance rate of draft tokens from the usual $\min\left(1, \frac{p(x)}{q(x)}\right)$ to $\min\left(1, \frac{p(x)}{aq(x)}\right)$.

In the case where a draft token is rejected, a replacement token is sampled from an ammended variant of the residual distribution incorporating the probability a

$$p_{res}(x;a) = norm(max(p(x) - aq(x), 0)). \quad (2)$$

In the $1-a$ proportion of cases where we do not draft — the *no-draft* case — we might naively expect that we can sample the next token directly from the base model. However, this would result in oversampling. In the no-draft case we must sample the next token from the residual distribution Eqn. 2 in order to ensure fidelity (see Figure 2). Superficially this suggests Randomised Drafting should provide no benefit since the residual distribution Eqn. 2 is a function of the draft distribution meaning that the draft model still needs to be invoked to sample the next token. However, the gain comes from the fact that, while the draft model needs to be employed for verfication it does not need to be employed for drafting. In practice this means that verification does not need to wait a draft token to be generated. I.e rather than sampling a draft token from q and then passing this into p for verfication we pass the existing prompt though both p and q in parallel, compute the residual distribution and sample the next token from it. This is depicted in Figure 1.

In vanilla SD there are *two* scenarios: 1) a draft is generated and accepted, or 2) a draft is generated and it is rejected whereafter a replacement is sampled from the residual distribution. In RD there are *three* scenarios: 1) a draft is gen-

erated and accepted, 2) a draft is generated and it is rejected whereafter a replacement is sampled from the residual distribution or 3) no draft is generated, a sample is drawn from the residual distribution. By clever selection of the probability $a \in [0, 1]$ RD can yield a higher Tokens-Per-Second (TPS).

Algorithm 2 Randomised Drafting SD

Input: Base model $p(x \mid z)$, draft model $q(x \mid z)$, prompt z, drafting probability a
Sample $u \sim [0, 1]$
if $u < a$ **then**
  Sample draft $\tilde{x} \sim q(x \mid z)$
  Compute acceptance probability $\alpha = \min\left(1, \frac{p(\tilde{x}|z)}{aq(\tilde{x}|z)}\right)$
  With probability α, set $x \leftarrow \tilde{x}$
  Otherwise, sample $x \sim p_{\text{res}}(x \mid z; a)$
else
  Sample $x \sim p_{\text{res}}(x \mid z; a)$ {As defined in Eqn. 2}
end if
Output: Sample x

4.1. Properties of SD with Randomised Drafting

The Total Variation (TV) distance between two probability vectors $\boldsymbol{p}, \boldsymbol{q}$ may be defined as

$$\mid \boldsymbol{p} - \boldsymbol{q} \mid_1 .$$

We define the following generalization of the Total Variation distance between probability vectors $\boldsymbol{p}, \boldsymbol{q}$ given some probability a;

$$TV_a(\boldsymbol{p}, \boldsymbol{q}) :=\mid \boldsymbol{p} - a\boldsymbol{q} \mid_1 +1 - a. \tag{3}$$

Note that setting $a = 1$ returns the standard TV between $\boldsymbol{p}$ and $\boldsymbol{q}$.

The acceptance rate for a token when using vanilla SD is known to be $1 - \frac{1}{2}TV(\boldsymbol{p}, \boldsymbol{q})$ where $\boldsymbol{p}, \boldsymbol{q}$ denote the next-token distributions given some prompt z for the base and draft models respectively. The acceptance rate for SD with Randomised Drafting may be written in terms of the generalised TV defined in Eqn. 4.

Lemma 4.1. *SD with Randomised Drafting at probability a has a token acceptance rate given by*

$$\frac{1}{a}\left(1 - \frac{1}{2}TV_a(\boldsymbol{p}, \boldsymbol{q})\right) = \frac{1}{2a}\left(1 + a - \mid \boldsymbol{p} - a\boldsymbol{q} \mid_1\right), \tag{4}$$

where $\boldsymbol{p}, \boldsymbol{q}$ denote the next-token distributions for the base and draft distributions respectively.

It is straightforward to show that Equation 4 is monotonically decreasing in a. This is to say that the acceptance rate of SD with RD for a drafted token is always higher than that of vanilla SD.

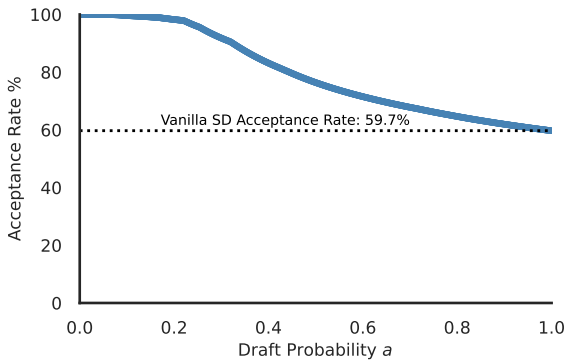

Figure 3. **Acceptance Rate as a function of the draft probability a:** We plot the acceptance rate Equation 4 as a function of a for a randomly selected prompt for OpenbookQA dataset. When $a = 1$ we get the acceptance rate for vanilla SD. As a decreases the acceptance probability increases. For $a < 0.2$ all drafted tokens would be accepted during verification.

Figure 3 plots the acceptance rate as a function of the draft probability for a randomly selected prompt drawn from the OpenbookQA (Mihaylov et al., 2018). I.e. given that a token has been drafted, we plot the probability that this token is accepted during verification. The base and draft model pair are the 6.7B and 2.7B parameters OPT models respectively. $a = 1$ (corresponding to vanilla SD) has an acceptance rate of $\approx 60\%$ for this prompt. As the draft probability decreases we see that the acceptance rate increases. In an extreme case where $a = 0.2$ all drafted tokens are accepted upon verfication.

RD has a strictly higher acceptance rate than vanilla SD when verfiying a draft token.

As discussed a core desirable property of vanilla SD is that it speeds up inference without altering the distribution of the output tokens, a property we refer to as *fidelity*. The following Lemma shows that this holds also for SD with Randomised Drafting.

Lemma 4.2. *Given a base model and draft model p, q respectively then, for any prompt z, samples obtained from SD with Randomised Drafting are distributed identically to those sampled directly from the base model $p(x \mid z)$.*

SD with RD satisfies Fidelity.

4.2. Impact of Random Drafting on TPS

We have shown that acceptance rate of a draft token when employing SD with RD is higher than vanilla SD. However,

this improved acceptance rate is offset by the fact that with probability $1 - a$ the TPS is equal to that of the base model. It is therefore non-trivial that RD could improve TPS over vanilla SD. This section derives a necessary and sufficient condition under which RD improves the TPS over vanilla SD. We then show how to derive the optimal choice of a.

For simplicity, we assume that verfiying and generating a token using the base model takes the same amount of time which we denote by T_p. We use T_q to denote the time taken to generate a token with the draft model q. We let λ denote the ratio of the time taken to generate a token with the draft and base model; $\lambda \coloneqq \frac{T_q}{T_p} (= \frac{\text{TPS}_p}{\text{TPS}_q})$. For example if, say, the draft model is twice as fast then $\lambda = 0.5$.

Lemma 4.3. *Given a prompt z let $\boldsymbol{p}, \boldsymbol{q}$ denote the next-token distributions of the base and draft model respectively. Randomised Decoding will improve the TPS if the following condition is satisifed*

$$\lambda \geq \sum_{i:p_i > q_i} q_i.$$

Lemma 4.3 states that if we sum up all those q_i for which the draft model underestimates the base model probability, then this should be less than the TPS ratio between the base and draft model. In practice this is often true.

Figure 4 plots the TPS as a function of the draft probability for the SD with RD when using the 6.7B and 2.7B parameter Facebook OPT models (as base/draft models) on the OpenbookQA dataset. We record that, on average the TPS of the the 6.7B parameter model is 0.6 times that of the smaller model $\lambda = 0.6$ in our setup. The y-axis is given relative to the TPS of the base model, thus $a = 0$ —the case where SD is not being applied— gives a TPS of 1x the base TPS as expected. $a = 1$ denotes the TPS gain obtained when using vanilla SD $\approx 8\%$. We observe that for some choices of $a < 1$ that the TPS is improved further $\approx 10\%$.

Randomised Drafting can improve TPS.

4.3. Choosing The Draft Probability

Our analysis so far reveals that the TPS may be enhanced by the use of random drafting with a suitable choice of a. Figure 4 shows an example of the relationship between the TPS and draft probability indicating that a value of $a \approx 0.8$ would be optimal in this particular setup. However, this optimal value of a is identified post factum. In practice we require an approach for selecting the draft probability *during* deployment on a workload rather than afterwards. This section proposes a practical approach for selecting a.

In the event that the condition given in Lemma 4.3 is satisfied we must determine a choice of $a \in [0, 1]$. The following Lemma describes how to find the optimal a given some

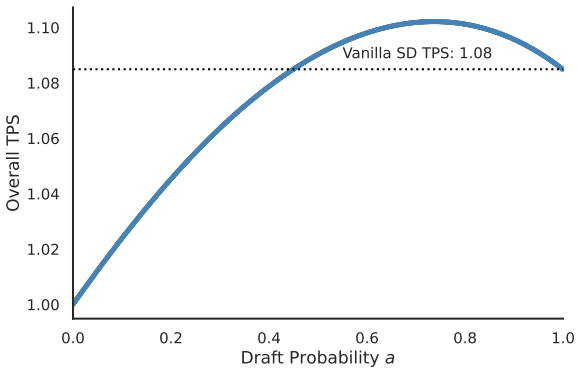

Figure 4. **TPS as a function of the draft probability a when applying SD with RD:** The y-axis is normalised relative to the base model so that 1.00 denote the TPS of the base model. We use the 6.7B/2.7B OPT models as the base/draft LLMs respectively and the OpenbookQA dataset of prompts for the dataset. We plot the TPS when applying SD with Randomised Drafting as a is varied. The resulting curve shows that the highest TPS is obtained for a value of $a < 1$ indicating an improvement over vanilla SD.

prompt z in the idealised setting where we have access to the next token distributions $\boldsymbol{p}$ and $\boldsymbol{q}$.

Lemma 4.4. *Given some prompt z, the value of a which optimizes the TPS of SD with Randomised Drafting is given by the solution to the following convex optimization problem:*

$$\underset{a \in [0,1]}{\textit{minimizing}} \quad |\boldsymbol{p} - a\boldsymbol{q}|_1 + a\,(2\lambda - 1)\,.$$

where $\boldsymbol{p}, \boldsymbol{q}$ denote next-token distributions for the base and draft models respectively given the prompt.

Equation 4.4 is a simple convex optimization problem meaning that, given $\boldsymbol{p}$ and $\boldsymbol{q}$, the optimal value $a^* \in [0, 1]$ maximising the TPS for the given prompt can be found efficiently using standard techniques. Unfortunately however, during inference p, q are not known ahead of time. Thus we have no way to precisely determine the optimal a for a given prompt in advance.

To resolve this problem, instead of finding an optimal a for each prompt (i.e. a depending on z) we drop the dependence on the prompt and instead determine a fixed a which maximises the TPS across a specific workload. Concretely, letting $r(z)$ denote some prompt distribution we wish to find $a \in [0, 1]$ which optimizes the following objective equivalent to optimizing the expected TPS on the workload

$$\mathbb{E}_{z \sim r(z)} \left[|\boldsymbol{p}(z) - a\boldsymbol{q}(z)|_1 \right] + a\,(2\lambda - 1)\,.$$

Crucially, given a dataset of samples $z_i \sim r(z)$ we can

estimate a^* by solving the following convex optimization

$$\underset{a\in[0,1]}{\text{minimizing}} \quad \left(\frac{1}{N}\sum_{i=1}^{N}|\boldsymbol{p}(z_i) - a\boldsymbol{q}(z_i)|_1\right) + a\,(2\lambda - 1)\,.$$

Concatenating the $\{\frac{1}{N}\boldsymbol{p}(z_i)\}_{i=1}^{N}$ vectors together into a single probability vector of length NV (and likewise for the $\boldsymbol{q}(z_i)$) we see that this problem is of the same form to the original optimization defined in Equation 4.4 and thus is similarly straightforward to solve.

Estimating the optimal draft probability for a given workload is a convex optimization problem.

5. Conclusions

This work introduces Speculative Decoding with Randomised Drafting. This entails drafting the next token with probability a. While we draft less frequently, RD allows us to accept a higher proportion of draft tokens than in the vanilla setting. For suitable choices of a, the boost in TPS obtained in the *draft* setting can be sufficient to offset the low TPS in the *no-draft* setting providing a gain over vanilla SD (See Figure 4). As with vanilla SD, applying Randomised Drafting generates outputs indistinguishable to those drawn from the base model directly. Moreover, applying RD does not increase the computational cost of applying SD - the total FLOP count remains identical to that of vanilla SD with the same base/draft model pair. The difference between these approaches being when the draft model is invoked (see Figure 1).

5.1. Limitations and Future Work

This research on randomised policies to Speculative Decoding is ongoing and significant further work is required. Further experimental analysis of the impact of RD on TPS in typical deployment scenarios is required. Additionally, further work should look at extending random drafting to draft chain lengths beyond $k = 1$. An example approach could be to determine the draft chain length by a randomised policy such as flipping a 'coin' until it comes up heads.

This work has limitations. For example, we demonstrate in Lemma 4.3 that the TPS ratio between the draft and base models λ should not be close to zero in order for RD to provide a gain. Intuitively this makes sense since the gain from RD comes from avoiding time uneccesarily spent drafting only for the drafted token to then be discarded. Therefore, when drafting is nearly instantaenous there is no gain to be obtained from opting not to draft. This could apply to settings where one uses an N-gram draft models or applies self-decoding techniques. Another possible limitation, highlighted by Figure 1, is that RD requires one to run the draft model in parallel with the base model. This is practical when the draft model is run on a separate device (e.g. a separate GPU to the base model or on the CPU) but may be impractical in some scenarios.

Impact Statement

"This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here."

References

Ankner, Z., Parthasarathy, R., Nrusimha, A., Rinard, C., Ragan-Kelley, J., and Brandon, W. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

Gloeckle, F., Idrissi, B. Y., Rozière, B., Lopez-Paz, D., and Synnaeve, G. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Hu, Y., Liu, Z., Dong, Z., Peng, T., McDanel, B., and Zhang, S. Q. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*, 2025.

Huang, K., Guo, X., and Wang, M. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *arXiv preprint arXiv:2405.19715*, 2024.

Kim, T., Suresh, A. T., Papineni, K., Riley, M. D., Kumar, S., and Benton, A. Accelerating blockwise parallel language models with draft refinement. *Advances in Neural Information Processing Systems*, 37:34294–34321, 2024.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-

v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Liu, T., Li, Y., Lv, Q., Liu, K., Zhu, J., and Hu, W. Parallel speculative decoding with adaptive draft length. *arXiv preprint arXiv:2408.11850*, 2024b.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Monea, G., Joulin, A., and Grave, E. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023.

Narasimhan, H., Jitkrittum, W., Rawat, A. S., Kim, S., Gupta, N., Menon, A. K., and Kumar, S. Faster cascades via speculative decoding. *arXiv preprint arXiv:2405.19261*, 2024.

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

Wang, J., Su, Y., Li, J., Xia, Q., Ye, Z., Duan, X., Wang, Z., and Zhang, M. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025.

Yang, S., Lee, G., Cho, J., Papailiopoulos, D., and Lee, K. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*, 2023.

A. Randomised Drafting Derivations

A.1. TPS Analysis

In the following we make some simplifying assumptions. For simplicity we assume that only a single token is being drafted by the draft model ($k = 1$). We assume that the time taken to verify and to decode a single token is the same when using the base model, denoting both by T_p. Similarly, we use T_q to denote the time taken to generate a draft token using the draft model. Letting α denote the acceptance rate for the given prompt. Then the expected TPS when using SD is

$$\text{TPS}_{SD} = \alpha \left(\frac{2}{T_p + T_q} \right) + (1 - \alpha) \left(\frac{1}{T_p + T_q} \right).$$

This expression reflects the fact that, with probability α we obtained 2 tokens in time $T_p + T_q$ and with probability $1 - \alpha$ we obtain just one.

For RD the expected TPS is given by

$$\text{TPS}_{RD} = a\alpha(a) \left(\frac{2}{T_p + T_q} \right) + a(1 - \alpha(a)) \left(\frac{1}{T_p + T_q} \right) + (1 - a) \frac{1}{T_p}$$

where $\alpha(a)$ denotes the acceptance rate of a draft token for the current prompt when drafting with probability a. This expression combines the case where we draft and the draft token is accepted, where we draft and it is rejected and where we do not draft. Our goal is to demonstrate that under certain scenarios that this second expression may be higher reflecting the fact that Randomised Drafting may boost ones TPS.

The difference in expected TPS between RD SD and vanilla SD can be written as

$$\text{TPS}_{RD} - \text{TPS}_{SD} = \frac{1}{T_p + T_q} \left(a\alpha(a) + a - \alpha - 1 \right) + \frac{1 - a}{T_p}.$$

We use λ to denote the ratio between T_p and T_q, that is $\lambda \coloneqq \frac{T_q}{T_p} \in [0, 1]$. By Equation 4

$$\begin{aligned}
\text{TPS}_{RD} - \text{TPS}_{SD} &= \frac{1}{T_p} \left(\frac{1}{1 + \lambda} (a\alpha(a) + a - \alpha - 1) + 1 - a \right), \\
&= \frac{1}{T_p} \left(\frac{1}{1 + \lambda} \left(\frac{1}{2} (1 + a - \mid p - aq \mid) + a - \alpha - 1 \right) + 1 - a \right), \\
&= \frac{1}{T_p} \left(\frac{1}{1 + \lambda} \left(\frac{3}{2} a - \frac{1}{2} - \frac{\mid p - aq \mid}{2} - \alpha \right) + 1 - a \right). \qquad (5)
\end{aligned}$$

This expression is concave since it is of the form $-k_1 \mid p - aq \mid + k_2 a + k_3$, and $\mid p - aq \mid$ is convex. We can evaluate at $a = 0$

$$\text{TPS}_{RD}(a = 0) - \text{TPS}_{SD} = 1 - \frac{\alpha + 1}{\lambda + 1}.$$

We assume that we are in a regime where SD improves the TPS thus $\alpha > \lambda$ which implies $1 - \frac{\alpha+1}{\lambda+1} < 0$. We evaluate at $a = 1$ to obtain

$$\text{TPS}_{RD}(a = 1) - \text{TPS}_{SD} = 0.$$

We wish to demonstrate that for some choices p, q that there are choices of $a \in [0, 1]$ for which $\text{TPS}_{RD}(a) - \text{TPS}_{SD} > 0$ indicating that the expected TPS of RD is greater than SD.

Since we have established that the difference in TPS is zero when $a = 1$ and the difference in TPS is concave in a this is equivalent to showing $\text{TPS}_{RD}(a) - \text{TPS}_{SD}$ may have a negative gradient at $a = 1$. Taking the derivative of Equation 5 with respect to a, this may be written as

$$\begin{aligned}
\frac{1}{T_p} \left(\frac{1}{1 + \lambda} \left(1 + \sum_i \mathbb{1}_{\{i : p_i > q_i\}} q_i \right) - 1 \right) &\leq 0, \\
\iff \sum_{i : p_i > q_i} q_i &\leq \lambda.
\end{aligned}$$

I.e, if we sum up all those q_i for which $p_i > q_i$ then this should be less than the ratio $\lambda \coloneqq \frac{T_q}{T_p}$. Thus we have demonstrated Lemma 4.3 restated below.

Lemma A.1. *Given a prompt z let $\boldsymbol{p}, \boldsymbol{q}$ denote the next-token distributions of the base and draft model respectively. Randomised Decoding will improve the TPS if the following condition is satisifed*

$$\lambda \geq \sum_{i:p_i>q_i} q_i.$$

Proof. Derived in the TPS analysis above. □

In general we may wish to identify the optimal choice of a. This is the value of a which maximises the expression in Equation 5. This boils down to solving the convex optimization problem:

$$\underset{a\in[0,1]}{\text{minimizing}} \quad |p - aq| + a\,(2\lambda - 1)\,.$$

Lemma A.2. *Given some prompt z, the value of a which optimizes the TPS of SD with Randomised Drafting is given by the solution to the following convex optimization problem:*

$$\underset{a\in[0,1]}{\textit{minimizing}} \quad |\boldsymbol{p} - a\boldsymbol{q}|_1 + a\,(2\lambda - 1)\,. \tag{6}$$

where $\boldsymbol{p}, \boldsymbol{q}$ denote next-token distributions for the base and draft models respectively given the prompt.

Proof. Equation 5 gives the TPS difference between RD and SD. We wish to maximise this with respect to $a \in [0, 1]$. One may discard constant terms and mutliply through by positive scalars without changing the optima. Finally, multiplying through by -1 yields the *minimization* problem given in Equation 6. □

Given p, q this optimization can easily be solved using standard approaches. A difficulty here is that we do not know ahead of time and thus the optimal a for a given prompt z can only be computed in retrospect.

Our proposed approach is to determine a by profiling a suitable respresentative workload. Specifically, our goal is finding $a^* \in [0, 1]$ which minimises the following expression

$$\mathbb{E}_{z\sim p(z)}\left[|p(z) - aq(z)|\right] + a\,(2\lambda - 1)\,,$$

where $p(z)$ denotes some distribution over prompts for which we desired the optimal TPS. Thus given samples $z_i \sim p(z)$ we can estimate a^* by solving

$$\underset{a\in[0,1]}{\text{minimizing}} \quad \left(\frac{1}{N}\sum_{i=1}^{N} |p(z_i) - aq(z_i)|\right) + a\,(2\lambda - 1)\,.$$

Lemma A.3. *SD with Randomised Drafting at probability a has a token acceptance rate given by*

$$\frac{1}{a}\left(1 - \frac{1}{2}TV_a(p, q)\right) = \frac{1}{2a}\left(1 + a - \mid p - aq \mid\right), \tag{7}$$

where p, q denote the token distributions for the base and draft distributions respectively.

Proof. The probability that the next token is accepted is simply

$$\begin{aligned}
\sum_{x\in\mathcal{X}} q(x)\min\left(1,\frac{p(x)}{aq(x)}\right) &= \sum_{x\in\mathcal{X}} q(x)\mathbb{1}_{\{aq(x)>p(x)\}} + \sum_{x\in\mathcal{X}} q(x)\mathbb{1}_{\{aq(x)\leq p(x)\}},\\
&= \sum_{x\in\mathcal{X}} q(x)\mathbb{1}_{\{aq(x)>p(x)\}}\frac{p(x)}{aq(x)} + \sum_{x\in\mathcal{X}} q(x)\mathbb{1}_{\{aq(x)\leq p(x)\}},\\
&= \sum_{x\in\mathcal{X}} \mathbb{1}_{\{aq(x)>p(x)\}}\frac{1}{a}p(x) + \sum_{x\in\mathcal{X}} q(x)\mathbb{1}_{\{aq(x)\leq p(x)\}},\\
&= \frac{1}{a}\left(\sum_{x\in\mathcal{X}} \mathbb{1}_{\{aq(x)>p(x)\}}p(x) + \sum_{x\in\mathcal{X}} aq(x)\mathbb{1}_{\{aq(x)\leq p(x)\}}\right), \qquad (8)\\
&= \frac{1}{a}\left(1 - \sum_{x\in\mathcal{X}} \mathbb{1}_{\{aq(x)\leq p(x)\}}(p(x)-aq(x))\right). \qquad (9)
\end{aligned}$$

Now,

$$\begin{aligned}
TV_a(p,q) &\coloneqq 1-a+\sum_{x\in\mathcal{X}} \mid p(x)-aq(x)\mid\\
&= 1-a+\sum_{x\in\mathcal{X}}(p(x)-aq(x))\mathbb{1}_{\{aq(x)\leq p(x)\}} + \sum_{x\in\mathcal{X}}(aq(x)-p(x))\mathbb{1}_{\{aq(x)>p(x)\}}\\
&= 1-a+\left(2\sum_{x\in\mathcal{X}}(p(x)\mathbb{1}_{\{aq(x)\leq p(x)\}})-1\right)+\left(a-2\sum_{x\in\mathcal{X}}(aq(x)\mathbb{1}_{\{aq(x)\leq p(x)\}})\right)\\
&= 2\sum_{x\in\mathcal{X}}(p(x)-aq(x))\mathbb{1}_{\{aq(x)\leq p(x)\}})
\end{aligned}$$

From which it follows that we can write Equation 8 as

$$\frac{1}{a}\left(1-\sum_{x\in\mathcal{X}} \mathbb{1}_{\{aq(x)\leq p(x)\}}(p(x)-aq(x))\right) = \frac{1}{a}\left(1-\frac{1}{2}TV_a(p,q)\right),$$

as desired. □

Lemma A.4. *Given a base model and draft model p, q respectively then, for any prompt z, samples obtained from SD with Randomised Drafting are distributed identically to those sampled directly from the base model $p(x \mid z)$.*

Proof. SD with Randomised Drafting generates samples distributed according to the base model p. For the purposes of this proof we will let $p(x_0)$ denote $p(x_0 \mid z)$ and $q(x_0)$ denote $q(x_0 \mid z)$ for notational brevity. Let x_0 be some arbitary token satisfying $aq(x_0) \leq p(x_0)$. The probability of sampling token x_0 when using Randomised Drafting can be written

$$\begin{aligned}
aq(x_0)\min\left(1,\frac{p(x_0)}{aq(x_0)}\right) + a(1-\alpha(z))p_{res,a}(x_0) + (1-a)p_{res,a}(x_0)&\\
= aq(x_0) + p_{res,a}(x_0)\left(a(1-\alpha(z)) + (1-a)\right). \qquad (10)&
\end{aligned}$$

We recall that

$$\begin{aligned}
p_{res,a}(x_0) &\coloneqq norm(max(p-aq,0))(x_0),\\
&= \frac{\max(p(x_0)-aq(x_0),0)}{\sum_x \max(p(x)-aq(x),0)},\\
&= \frac{p(x_0)-aq(x_0)}{\frac{1}{2}TV_a(p,q)}.
\end{aligned}$$

So, Equation 10 is equal to

$$aq(x_0) + (1 - a\alpha(z)) \frac{p(x_0) - aq(x_0)}{\frac{1}{2}TV_a(p,q)}. \tag{11}$$

We recall that $\alpha(a) = \frac{1}{a}\left(1 - \frac{1}{2}TV_a(p,q)\right)$ from which it follows that Equation 11 is equal to

$$aq(x_0) + \left(\frac{1}{2}TV_a(p,q)\right) \frac{p(x_0) - aq(x_0)}{\frac{1}{2}TV_a(p,q)} = p(x_0).$$

Now let x_0 be an arbitrary token such that $aq(x_0) \geq p(x_0)$. The probability of sampling token x_0 when using SD with Randomised Drafting can be written

$$\begin{aligned} aq(x_0) \min\left(1, \frac{p(x_0)}{aq(x_0)}\right) + a(1 - \alpha(z))p_{res,a}(x_0) + (1-a)p_{res,a}(x_0) \\ = p(x_0) + p_{res,a}(x_0)\left(a(1-\alpha(z)) + (1-a)\right). \end{aligned} \tag{12}$$

However, $p_{res,a}(x_0) = 0$ thus Equation 12 is simply equal to $p(x_0)$. Consequently, we have demonstrated that the probability of sampling each token $x \in \mathcal{X}$ when applying RD is equal to the probability of sampling this token with the base model $p(x)$. □