Speculative Decoding

Karl Stratos

1 Speculative Sampling

Let p be a distribution over $x \in \mathcal{X}$. Let $q \neq p$ be a proposal distribution. We want to design a "speculative" distribution s that (i) first samples $x \sim q$, (ii) accepts or rejects x, and (iii) re-samples x in the case of reject, such that $s = p$. We want to accept $x \sim q$ as frequently as possible (e.g., a bad solution is to always reject and resample $x \sim p$). By the generative process, the probability that s assigns on each $x \in \mathcal{X}$ is

$$s(x) = \Pr(x \text{ is accepted}) \times q(x) + \Pr(\text{some sample from } q \text{ is rejected}) \times \mu(x)$$

where μ is a re-sampling distribution. By the constraint $s(x) = p(x)$, μ is uniquely identified by

$$\mu(x) \propto p(x) - \Pr(x \text{ is accepted}) \times q(x) \tag{1}$$

A reasonable policy is to accept $x \sim q$ iff $p(x) \geq q(x)$, but it rejects all x such that $p(x) < q(x)$. To do better, we can *stochastically* accept such x with probability $\frac{p(x)}{q(x)} < 1$. Then $\Pr(x \text{ is accepted}) = \min(1, \frac{p(x)}{q(x)})$ and (1) is simplified as

$$\mu(x) \propto \max(p(x) - q(x), 0)$$

In summary, we may sample $x \sim p$ through q as follows:

Input: p, q over $\mathcal{X}$
Output: $x \sim p$

1. Sample $x \sim q$.
2. Sample $z \sim \text{Ber}(\min(1, \frac{p(x)}{q(x)}))$.
3. If $z = 1$, return x.
4. For each $y \in \mathcal{X}$, set $\mu(y) \leftarrow \max(p(y) - q(y), 0)$. Renormalize μ.
5. Return $x \sim \mu$.

1.1 Speculative Decoding

Concurrent works [3, 2] proposed an autoregressive extension for fast sampling from a language model.

Input: p, q over $\mathcal{X}^+$, length T
Output: autoregressive sampling $x_1 \dots x_L \sim p$ for some $1 \leq L \leq T + 1$

1. Sample $x_1 \dots x_T \sim q$ autoregressively.
2. Compute $p(x_t|x_{<t})$ for all t in parallel (by causal masking).
3. Sample $z_t \sim \text{Ber}(\min(1, \frac{p(x_t|x_{<t})}{q(x_t|x_{<t})}))$ for all t in parallel.
4. If $z_t = 1$ for all t, return $x_1 \dots x_T$ and $x_{T+1} \sim p(\cdot|x_{\leq T})$.
5. Let $n \leftarrow \min\{1 \leq t \leq T : \ z_t = 0\}$.
6. For each $y \in \mathcal{X}$, set $\mu(y) \leftarrow \max(p(y|x_{<n}) - q(y|x_{<n}), 0)$. Renormalize μ.
7. Return $x_1 \dots x_n$ and $x_{n+1} \sim \mu$.

Here, p is a language model possibly conditioned on some context and q is a small "draft" language model (also conditioned on the same context). The key ideas are (i) we can compute p's probabilities for all autoregressively generated tokens from q in one shot (Step 2) and (ii) we can retain all consecutively accepted tokens without loss of correctness (Step 5). These ideas have previously been used for greedy decoding [6] (Appendix A).

1.1.1 Length analysis

The acceptance probabilities across steps are dependent random variables: $\beta_t = \mathbf{E}_{x \sim q(\cdot|x_{<t})}[\min(1, \frac{p(x|x_{<t})}{q(x|x_{<t})})] = \sum_{x \in \mathcal{X}} \min(q(x|x_{<t}), p(x|x_{<t}))$ where $x_1 \ldots x_T \sim q$. For analysis, however, assume that they are are iid (e.g., p, q are unigram language models) and set β as their mean. Then L is a capped geometric variable with success probability $1 - \beta$ and max length $T + 1$. We have[1]

$$\mathbf{E}[L] = \frac{1 - \beta^{T+1}}{1 - \beta}$$

For instance, if we accept at each step with probability $\beta = 0.8$ and set $T = 10$, we can expect to generate > 4 tokens in a single step of speculative decoding.

2 Iterative Speculative Sampling

We can do speculative sampling iteratively. Let p, q, q' be distributions over $\mathcal{X}$. If

$$p(x) = \Pr(x \text{ is accepted}) \times q(x) + \Pr(\text{some sample from } q \text{ is rejected}) \times \mu(x)$$

we can again independently reparameterize

$$\mu(x) = \Pr(x \text{ is accepted}) \times q'(x) + \Pr(\text{some sample from } q' \text{ is rejected}) \times \mu'(x)$$

Here is an iterative version of speculative sampling in Section 1. We sample from a proposal distribution K times before resorting to p. Note that we do not require $q_k \neq q_{k'}$.

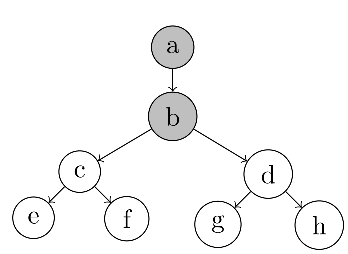
Input: $p, q_1 \ldots q_K$ over $\mathcal{X}$
Output: $x \sim p$

1. $p_0 \leftarrow p$
2. For $k = 1 \ldots K$:
   (a) Sample $x \sim q_k$.
   (b) Sample $z \sim \mathrm{Ber}(\min(1, \frac{p_{k-1}(x)}{q_k(x)}))$.
   (c) If $z = 1$, return x.
   (d) For each $y \in \mathcal{X}$, set $p_k(y) \leftarrow \max(p_{k-1}(y) - q_k(y), 0)$. Renormalize p_k.
3. Return $x \sim p_K$.

2.1 SpecInfer

SpecInfer [4] trains K draft models $q_1 \ldots q_K$ to cover the top predictions of p.[2] Given any context, they generate K drafts of length T in parallel. We can economically compute all of p's conditional distributions by "tree attention": pack the drafts into a prefix tree and perform tree masking on tokens ordered by depth-first search. An example with $K = 4$ drafts of length $T = 2$ with context $(a, b) \in \mathcal{X}^2$ is

t	-1	0	1	2
$x^{(1)}$	a	b	c	e
$x^{(2)}$	a	b	c	f
$x^{(3)}$	a	b	d	g
$x^{(4)}$	a	b	d	h



$$\begin{array}{c|cccccccc} & a & b & c & e & f & d & g & h \\ \hline a & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ b & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ c & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ e & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ f & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ d & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ g & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ h & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}$$

We can run the model on (a, b, c, e, f, d, g, h) (where the hidden states of a, b are in the KV cache) with tree masking to compute $p(\cdot|x^{(k)}_{\leq t})$ for all $k = 1, 2, 3, 4$ and $t = 0, 1, 2$. This involves fewer tokens than batched causal masking.

[1] Deriving this formula requires a nontrivial analysis of $\mathbf{E}[L] = \sum_{t=1}^{T+1} (1 - \beta)\beta^{t-1} t$.

[2] Specifically, they are trained in sequence where the k-th dataset excludes examples such that p and any previous $q_1 \ldots q_{k-1}$ predict the same output.

Input: $p, q_1 \dots q_K$ over $\mathcal{X}^+$, context $u = (u_{-M} \dots u_0)$, length T
Output: autoregressive sampling $x_1 \dots x_L \sim p(\cdot|u)$ for some $1 \leq L \leq T+1$

1. For all k in parallel: sample $x_1^{(k)} \dots x_T^{(k)} \sim q_k(\cdot|u)$ autoregressively. Denote $x_j^{(k)} = u_j$ for $j = -M \dots 0$.
2. Build a prefix tree and run the tree attention. Let ν_0 denote the node corresponding to u_0, whose t-th descendent ν has $\nu.x = x_t^{(k)}$, $\nu.q = q_k(\cdot|x_{<t}^{(k)})$, and $\nu.p = p(\cdot|x_{\leq t}^{(k)})$ for some k.
3. $\nu \leftarrow \nu_0$, $V \leftarrow []$
4. While ν is not a leaf node:
   (a) For each child γ of ν in random order:
      i. Sample $z \sim \text{Ber}(\min(1, \frac{\nu.p(\gamma.x)}{\gamma.q(\gamma.x)}))$.
      ii. If $z = 1$, set $V \leftarrow V + [\gamma.x]$, $\nu \leftarrow \gamma$, and continue Loop 4.
      iii. Else, update $\nu.p(y) \propto \max(\nu.p(y) - \gamma.q(y), 0)$.
   (b) Go to Step 5.
5. Sample $x \sim \nu.p$ and return $V \leftarrow V + [x]$.

Correctness: At any point, V is some valid partial draft(s). The children of ν are valid continuations from some proposal distributions. We apply the iterative speculative sampling to get another valid V.

3 Draft Models

The draft model must be both fast and similar to the main model. For instance, we may use a small model in a family of pretrained models to draft for a big one (e.g., 8B for 70B Llama-3). To avoid running multiple models, many works consider making the main model draft for itself. In particular, it can be trained to predict several future tokens in parallel (aka. semi-autoregressive). A simple example is PaSS [5] which trains T "look-ahead" embeddings $[\text{LA}]_1 \dots [\text{LA}]_T$ by masking the last T tokens of any observed sequence $x_1 \dots x_m, x_{m+1} \dots x_{m+T+1} \in \mathcal{X}$ and maximizing the likelihood $p(x_{m+t}|x_1 \dots x_m, [\text{LA}]_{<t})$. At test time, it appends $[\text{LA}]_1 \dots [\text{LA}]_T$ to any context $x_1 \dots x_m$ and samples the next $T+1$ tokens $x_{m+1} \dots x_{m+T+1}$ in parallel. These tokens are verified as usual by treating the look-ahead conditional as a proposal distribution, i.e., for $t = 1 \dots T+1$, sample

$$z_t \sim \text{Ber}\left(\min\left(1, \frac{p(x_{m+t}|x_1 \dots x_m, x_{m+1} \dots x_{m+t-1})}{p(x_{m+t}|x_1 \dots x_m, [\text{LA}]_1 \dots [\text{LA}]_{t-1})} =: \frac{P(x_{m+t})}{Q(x_{m+t})}\right)\right)$$

and accept x_{m+t} if $z_t = 1$, else re-sample x_{m+t} from a renormalized distribution $\propto \max(P(\cdot) - Q(\cdot), 0)$ and return $x_{m+1} \dots x_{m+t}$. Blockwise parallel decoding trains T separate decoding layers [6]. Medusa [1] combines these techniques to optimize performance. It trains five additional decoding layers to sample the next six tokens in parallel, then verifies a promising subset of speculated sequences with the tree attention on a top-k prefix tree. It also proposes fine-tuning the backbone model itself in addition to the decoding layers (Medusa-2), which further increases the inference speed at the cost of changing the original model.

References

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

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

[3] Leviathan, Y., Kalman, M., and Matias, Y. (2023). Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

[4] Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R. Y. Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. (2024). 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*, ASPLOS '24, page 932–949, New York, NY, USA. Association for Computing Machinery.

[5] Monea, G., Joulin, A., and Grave, E. (2023). Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*.

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

A Greedy Speculative Decoding

By $(x_1 \dots x_T) \leftarrow \text{Greedy}(p)$, we mean autoregressively predicting $x_t = \arg\max_{x \in \mathcal{X}} \; p(x|x_{<t})$. Here is a greedy version of the speculative decoding in Section 1.1.

Input: p over $\mathcal{X}^+$, speculated sequence $x_1 \dots x_T \in \mathcal{X}$
Output: $(x_1 \dots x_L) \leftarrow \text{Greedy}(p)$ for some $1 \leq L \leq T+1$

1. Predict $y_t \leftarrow \arg\max_{y \in \mathcal{X}} p(y|x_{<t})$ for all $1 \leq t \leq T+1$ in parallel (by causal masking).
2. If $y_t = x_t$ for all $1 \leq t \leq T$, return $(x_1 \dots x_T, y_{T+1})$.
3. Let $n \leftarrow \min \{1 \leq t \leq T : \; y_t \neq x_t\}$.
4. Return $(x_1 \dots x_n, y_{n+1})$.

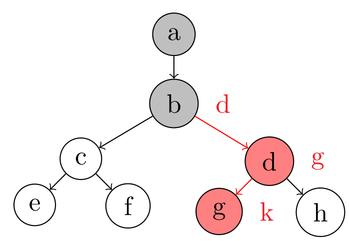
A naive extension to multiple speculated sequences is straightforward: we do the above procedure in parallel by batching, and pick the longest output. Alternatively, we can again use the tree attention in Section 2.1 to perform greedy decoding more economically:

Input: p over $\mathcal{X}^+$, K speculated sequences $x^{(1)} \dots x^{(K)}$ with a shared context $x_j^{(k)} = u_j$ for $j = -M \dots 0$
Output: $(x_1 \dots x_L) \leftarrow \text{Greedy}(p(\cdot|u_{-M} \dots u_0))$ for some $1 \leq L \leq T+1$

1. Build a prefix tree from $x^{(1)} \dots x^{(K)}$ and run the tree attention. Let ν_0 denote the node corresponding to u_0, whose t-th descendent ν has $\nu.x = x_t^{(k)}$ and $\nu.y = \arg\max_{y \in \mathcal{X}} p(y|x_{\leq t}^{(k)})$ for some k.
2. $\nu \leftarrow \nu_0$, $V \leftarrow []$
3. While ν is not a leaf node:
   (a) If some child γ of ν satisfies $\gamma.x = \nu.y$: $V \leftarrow V + [\gamma.x]$, $\nu \leftarrow \gamma$, continue Loop 3.
   (b) Go to Step 4.
4. Return $V \leftarrow V + [\nu.x]$.

Committing to some child at each node actually finds the longest accepted sequence because p's greedy decoding is deterministic given the same context. Thus at any node in the prefix tree, there is only one acceptable child or none. See the example below.

a	b (d)	c (e)	e (q)
a	b (d)	c (e)	f (l)
a	b (d)	d (g)	g (k)
a	b (d)	d (g)	h (m)



$V = [\text{d}, \text{g}, \text{k}]$