

Lecture 9: Chain-of-Thought and Speculative Decoding

Prof. Noah Golowich

Today we will study how chain-of-thought supports sequential computation and how speculative decoding speeds up generation.

1 Chain-of-thought

An autoregressive model can generate intermediate computations before answering a question. Each result becomes part of the context for the next prediction.

1.1 Chain-of-thought prompting

A *chain-of-thought* (CoT) is a sequence of intermediate reasoning steps generated before an answer. If x is a question, r a rationale, and y an answer, the model generates the concatenation (r, y) conditioned on x . The answer can therefore depend on the generated rationale:

$$p_{\theta}(r, y \mid x) = p_{\theta}(r \mid x) p_{\theta}(y \mid x, r).$$

Each factor is itself the product of next-token probabilities along the corresponding sequence. Here θ denotes the model parameters.

Wei et al. [1] include a few *question*, *rationale*, *answer* triples in the prompt, then append a new question. The demonstrations encourage the model to generate a rationale before answering. The model's parameters stay fixed. In standard few-shot prompting, the demonstrations contain only question–answer pairs.

Example. Consider a prompt with one worked example:

Q: A box contains 4 pencils. Maya adds 3 packs of 2 pencils each. How many pencils are in the box?

A: The packs add $3 \cdot 2 = 6$ pencils. Together with the original 4, this gives $4 + 6 = 10$. The answer is 10.

Q: A shelf holds 7 books. Leo adds 2 stacks of 5 books each, then lends out 3 books. How many books remain?

A:

A possible continuation is:

The stacks add $2 \cdot 5 = 10$ books, giving $7 + 10 = 17$. Lending out 3 leaves $17 - 3 = 14$. The answer is 14.

The model can now use the intermediate value 17 when generating the final subtraction. These extra steps give it more computation.

1.2 Learning rationales: STaR

Given a few worked examples and a larger dataset of questions with known answers, the Self-Taught Reasoner (STaR) method builds a training set of model-generated rationales (i.e., chains-of-thought) [2].

Rationale generation. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ be a dataset of questions and correct answers. Include a small, fixed set of rationale demonstrations in each x_i , suppressing this shared prompt in the notation. Starting with a pretrained model, repeat:

1. Generate a rationale $\hat{r}_i$ followed by an answer $\hat{y}_i$ for each x_i .
2. Keep $(x_i, \hat{r}_i, y_i)$ exactly when $\hat{y}_i = y_i$.
3. Fine-tune on these retained rationale–answer sequences, then use the resulting model to generate the next round’s data.

Each round of STaR fine-tunes the original pretrained model on the newly collected dataset. The filter checks only the final answer, so it can retain rationales with incorrect intermediate steps.

The policy-gradient interpretation. Treating the rationale as a latent variable, we can write the answer probability as

$$p_\theta(y \mid x) = \sum_r p_\theta(r \mid x) p_\theta(y \mid x, r).$$

Treat the generated rationale and answer as a sampled action sequence, with reward $\mathbf{1}\{\hat{y}_i = y_i\}$. The expected total reward is

$$J(\theta) = \sum_{i=1}^m \mathbb{E}_{(\hat{r}_i, \hat{y}_i) \sim p_\theta(\cdot, \cdot \mid x_i)} [\mathbf{1}\{\hat{y}_i = y_i\}]. \quad (1)$$

Its gradient is

$$\nabla_\theta J(\theta) = \sum_{i=1}^m \mathbb{E}_{(\hat{r}_i, \hat{y}_i) \sim p_\theta(\cdot, \cdot \mid x_i)} [\mathbf{1}\{\hat{y}_i = y_i\} \nabla_\theta \log p_\theta(\hat{r}_i, \hat{y}_i \mid x_i)]. \quad (2)$$

Equations (1)–(2) give the REINFORCE interpretation of STaR [2, Eqs. (1)–(2)]. We obtain the gradient by expanding the expectation and applying the log-derivative identity:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{i=1}^m \sum_{r, y} \mathbf{1}\{y = y_i\} \nabla_\theta p_\theta(r, y \mid x_i) \\ &= \sum_{i=1}^m \sum_{r, y} p_\theta(r, y \mid x_i) \mathbf{1}\{y = y_i\} \nabla_\theta \log p_\theta(r, y \mid x_i). \end{aligned}$$

An incorrect answer contributes zero gradient. A correct answer contributes the log-likelihood gradient for its entire rationale and answer, which motivates filtering successful samples and training on them to increase J .

The identity assumes samples from the current policy. STaR instead uses greedy generations and takes multiple fine-tuning steps on the same retained dataset, starting each round from the base model. Its updates therefore approximate the policy-gradient procedure and need not be unbiased REINFORCE estimates.

Rationalization. A question the model answers incorrectly contributes no training example. To get a rationale for such a question, STaR adds the correct answer y_i to the input as a hint and asks the model to try again. If the new answer matches y_i , add $(x_i, \hat{r}_i^{\text{rat}}, y_i)$ to the fine-tuning set, removing the hint from the input. These examples train the model to generate a rationale and answer from the question alone, even for some questions it initially answered incorrectly. Because generation uses the answer hint, these sequences do not come from the unhinted policy in (2).

1.3 Why intermediate steps can help: function composition

A natural setting where intermediate steps can provably help is to perform *function composition*. In particular, to compute $f(g(a))$, we first find $g(a)$, then use that value to query f . A single attention layer has limited ability to perform these two dependent lookups. We will make this limitation precise in the model of Peng, Narayanan, and Papadimitriou [3].

The transformer-layer model. We generally stick to our notation from Lecture 3, taking the input, query/key, value, and output dimensions all equal to d . The input representations form $X = [x_1, \dots, x_N] \in \mathbb{R}^{d \times N}$. Let H be the number of heads. For head $h \in [H]$, set

$$q_i^{(h)} = W_Q^{(h)} x_i, \quad k_j^{(h)} = W_K^{(h)} x_j, \quad v_j^{(h)} = W_V^{(h)} x_j,$$

where $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d \times d}$. Its output at position i is

$$y_i^{(h)} = \sum_{j=1}^N A_{ji}^{(h)} v_j^{(h)}, \quad A_{ji}^{(h)} = \frac{\exp(\langle q_i^{(h)}, k_j^{(h)} \rangle / \sqrt{d})}{\sum_{u=1}^N \exp(\langle q_i^{(h)}, k_u^{(h)} \rangle / \sqrt{d})}. \quad (3)$$

The layer applies a position-wise combining map

$$\Phi : (\mathbb{R}^d)^H \rightarrow \mathbb{R}^d, \quad x_i^{\text{out}} = \Phi(y_i^{(1)}, \dots, y_i^{(H)}).$$

A local decoder converts x_i^{out} into an answer.

Compared with the full block in Lecture 3, this model makes the following choices:

- The local postprocessing map Φ is arbitrary. It can include the output projection W_O , the position-wise multilayer perceptron (MLP), and post-attention normalization, or more general local computations. To include a residual connection, also give Φ the local input x_i . The decoder in our proof already knows this input at the query position.
- We can include token embeddings, positional information, and position-wise preprocessing in the x_j 's. This preprocessing must be local to each position, so it cannot include the output of a previous attention layer.
- The computation has finite precision. Write b for the number of bits per scalar.

The function composition problem. Let $[n] = \{1, \dots, n\}$, where $n \geq 2$. The prompt contains the tables of two arbitrary functions $g, f : [n] \rightarrow [n]$, followed by a query $a \in [n]$. The required output is $f(g(a))$ at a designated query position t . Use a fixed input layout: the g -tokens depend only on g , the f -tokens only on f , and the query tokens only on a . For example, a table can store each pair $(j, f(j))$ in a separate token. All model parameters and the layout are fixed independently of the tables.

For example, in “What is the profession of Alice’s mother?”, g maps a person to their mother and f maps a person to their profession. The query identifies Alice; the model must look up her mother’s identity.

Throughout the lower-bound argument, $\log$ has base two.

Theorem 1.1 (Single-layer composition lower bound [3, Theorem 1]). *In the model above, put $B = H(d+1)b$ and assume $B < n \log n$. If f and g are independently uniform among all maps $[n] \rightarrow [n]$, then for every fixed $a \in [n]$ the error probability is at least*

$$\frac{R}{3n \log n}, \quad R := n \log n - B > 0. \quad (4)$$

Consequently the same bound holds for an independent uniformly random query a .

We prove the theorem by reducing it to a communication problem: a sender who knows f must send a short message before learning which entry $g(a)$ will be requested. Fano's inequality will convert a bound on the information in the message into a lower bound on the error probability.

Fano's inequality. For a discrete random variable U , write $H(U) = -\sum_u \mathbb{P}(U = u) \log \mathbb{P}(U = u)$ for its entropy in bits; conditional entropy is the average entropy after conditioning. Write $h_2(u) = -u \log u - (1-u) \log(1-u)$ for binary entropy, with $0 \log 0 = 0$. We use

$$I(U; V \mid W) = H(U \mid W) - H(U \mid V, W)$$

for conditional mutual information, omitting W for ordinary mutual information. Fano's inequality bounds the uncertainty remaining about a value that can be estimated with small error.

Lemma 1.2 (Fano's inequality). *Let Y take values in $[n]$, where $n \geq 2$, and let Z be an observation. For any estimator $\hat{Y} = \psi(Z) \in [n]$ with error probability $\delta = \mathbb{P}(\hat{Y} \neq Y)$,*

$$H(Y \mid Z) \leq h_2(\delta) + \delta \log(n-1). \quad (5)$$

Proof. Let $E = \mathbf{1}\{\hat{Y} \neq Y\}$. Since E is determined by (Y, Z) , the entropy chain rule gives

$$H(Y \mid Z) = H(E \mid Z) + H(Y \mid E, Z) \leq h_2(\delta) + \delta \log(n-1).$$

Indeed, $H(E \mid Z) \leq H(E) = h_2(\delta)$. Given Z , the value of Y is determined when $E = 0$ and has at most $n-1$ possibilities when $E = 1$. $\square$

Lemma 1.3 (Communication complexity of indexing [3, Lemma 1]). *Faye knows $f : [n] \rightarrow [n]$ and sends a B -bit message to Xavier (at most 2^B possible messages). Xavier then receives an index $J \in [n]$ and must output $f(J)$; Faye does not know J when sending her message. If f is uniform and J is independently uniform, the error probability is at least*

$$\frac{n \log n - B}{3n \log n}.$$

The bound also holds if Xavier has additional information independent of f conditional on J .

Proof. Let $F = (f(1), \dots, f(n))$ be uniform on $[n]^n$, let $M = M(F)$ be the B -bit message, and let J be independently uniform on $[n]$. Fix a deterministic decoder $\hat{Y}(M, J)$ with error probability δ , and set $R = n \log n - B$. We may assume $R > 0$, since the claimed error bound is otherwise trivial.

Write $F_{<J} = (F_1, \dots, F_{J-1})$. Independence of the table entries gives $H(F_J) = H(F_J \mid F_{<J})$, while conditioning reduces entropy. Thus $I(M; F_J) \leq I(M; F_J \mid F_{<J})$, and the chain rule yields

$$\begin{aligned} I(M; F_J \mid J) &= \frac{1}{n} \sum_{j=1}^n I(M; F_j) \leq \frac{1}{n} \sum_{j=1}^n I(M; F_j \mid F_{<j}) \\ &= \frac{1}{n} I(M; F) \leq \frac{H(M)}{n} \leq \frac{B}{n}. \end{aligned}$$

Apply Fano's inequality with $Y = F_J$, observation $Z = (M, J)$, and estimator $\hat{Y}(M, J)$. Since $H(F_J | J) = \log n$, we obtain

$$\frac{R}{n} \leq \log n - I(M; F_J | J) = H(F_J | M, J) \stackrel{\text{Fano}}{\leq} h_2(\delta) + \delta \log(n-1). \quad (6)$$

The final inequality in (6), marked “Fano”, is precisely Lemma 1.2.

To obtain $\delta \geq R/(3n \log n)$, first suppose $\delta \geq 3/(4n)$. Using $h_2(u) \leq u \log(e/u)$ gives

$$\frac{R}{n} \leq \delta \log\left(\frac{e(n-1)}{\delta}\right) \leq 3\delta \log n,$$

where e is Euler's number and the last step uses $e(n-1)/\delta \leq (4e/3)n(n-1) \leq n^3$ for $n \geq 2$.

If $\delta < 3/(4n)$, a short counting argument completes the bound. Each message m determines a table $\hat{F}(m) = (\hat{Y}(m, 1), \dots, \hat{Y}(m, n))$, so at most 2^B of the n^n tables can be recovered exactly. An incorrect table has at least one incorrect entry, giving

$$n\delta \geq \mathbb{P}(F \neq \hat{F}(M)) \geq 1 - \frac{2^B}{n^n} = 1 - 2^{-R}.$$

Since $n\delta < 3/4$, we have $R < 2$. Concavity gives $1 - 2^{-R} \geq 3R/8$ on $[0, 2]$ (the chord between the endpoints). Therefore $\delta \geq 3R/(8n) \geq R/(3n \log n)$.

Side information independent of F conditional on J cannot improve on a most likely answer given (M, J) . For a randomized protocol, fix its random coins, apply the linear bound, and average. $\square$

Appendix A gives an alternative proof using sets of possible tables.

Proof of Theorem 1.1. Fix an input index $a \in [n]$. The *query* is the block of prompt tokens asking “What is $f(g(a))$?” Its wording and encoding follow a fixed template, so this block is determined by a and contains no information about the values of f or g . Faye receives the table of f , Grace receives the table of g , and Xavier initially receives neither table. All three know a , the query template, the input layout, and the model parameters; hence all three can construct the query tokens. There is no communication between Faye and Grace, and neither receives a message from Xavier. Grace may send Xavier all of g for free. We will simulate the layer with a message of B bits from Faye.

Partition the input positions into I_f, I_g, I_a , corresponding to the two tables and the query block. Let $t \in I_a$ be the designated position at which the layer must output the answer. For each head h , the *attention query vector* is $q_t^{(h)} = W_Q^{(h)} x_t$. Since the input representation x_t is determined by the query block, this vector depends only on a and the publicly known template and model parameters. In particular, Faye can compute it without knowing $g(a)$. Define the unnormalized weight $w_j^{(h)} = \exp(\langle q_t^{(h)}, k_j^{(h)} \rangle / \sqrt{d})$. Faye computes the partial numerator and denominator

$$S_f^{(h)} = \sum_{j \in I_f} w_j^{(h)} v_j^{(h)} \in \mathbb{R}^d, \quad Z_f^{(h)} = \sum_{j \in I_f} w_j^{(h)} \in \mathbb{R}. \quad (7)$$

Faye can compute these quantities from f and the public input index a . Sending them for all H heads takes $H(d+1)b = B$ bits. Xavier can compute the analogous quantities $(S_g^{(h)}, Z_g^{(h)})$ and $(S_a^{(h)}, Z_a^{(h)})$ himself, since he knows g and a . He reconstructs

$$y_t^{(h)} = \frac{S_f^{(h)} + S_g^{(h)} + S_a^{(h)}}{Z_f^{(h)} + Z_g^{(h)} + Z_a^{(h)}},$$

then applies Φ and the answer decoder. This reproduces the layer's prediction.

For uniform g , the index $J = g(a)$ is uniform on $[n]$ and independent of f . Faye's message does not depend on g and hence does not depend on J . Xavier's knowledge of the rest of g provides no information about f , even after conditioning on J . Lemma 1.3 therefore gives the error bound (4). $\square$

The obstruction is that all heads query the original representations in parallel: one head cannot use another head's discovery of $g(a)$ to choose its query. With a stack of layers, a later layer can use an earlier layer's output, so this argument no longer applies.

Iterated composition and CoT. For functions $f_1, \dots, f_K : [n] \rightarrow [n]$ and starting point a , define

$$z_0 = a, \quad z_j = f_j(z_{j-1}) \quad (1 \leq j \leq K).$$

The iterated composition problem asks for z_K . A special case has $f_1 = \dots = f_K = f$, so the answer is the K -fold iterate $f^{\circ K}(a)$.

Each CoT step runs the layer on the current prompt and appends a token that later steps can attend to. Generating a CoT therefore applies the same layer repeatedly in sequence.

Theorem 1.4 (Iterated composition [3, Theorem 2]). *In the finite-precision transformer-layer model, solving general iterated function composition on a domain of size n requires, in the worst case,*

$$\Omega\left(\sqrt{\frac{n}{Hdb}}\right)$$

CoT steps. Here the number of function applications is allowed to grow with the problem size; the claim is not a lower bound of this size for every fixed K .

We state this theorem without proof. The proof uses the communication complexity of *pointer chasing*, in which each function value determines the next table entry to access.

Sufficiently long CoT can carry out the iteration directly: write $z_1 = f_1(a)$, then $z_2 = f_2(z_1)$, and continue to z_K . Each step needs only one table lookup. For example, encode a table row's key (j, u) by a distinct binary sign vector. Its inner product with the matching query is strictly larger than with any other key. Scaling the scores makes the matching row receive attention weight greater than $1/2$. If values are binary encodings of $f_j(u)$, thresholding each coordinate at $1/2$ then recovers the value exactly. With sufficient precision and a token encoding the current value and stage, Φ can emit the state for the next lookup. Thus K sequential lookup steps suffice.

2 Speculative decoding

A difficult generation task can still contain easy tokens: completing a familiar phrase may take much less computation than choosing the next step of a mathematical argument. Speculative decoding lets a cheap model propose several tokens, then has an expensive model check them together. An acceptance and correction rule preserves the expensive model's sampling distribution [4].

2.1 Drafting and verification

Let $\mathcal{V}$ be a finite vocabulary. As in Lecture 3, $v_{<t} = (v_1, \dots, v_{t-1})$ denotes a token prefix. Let

$$p(v_t \mid v_{<t}) \quad \text{and} \quad q(v_t \mid v_{<t})$$

be the next-token distributions of an expensive *target model* and a cheaper *draft model*, respectively. We want to sample from the autoregressive distribution defined by p .

Fix a current prefix s and a draft length $\gamma \geq 1$. First sample draft tokens $\tilde{v}_1, \dots, \tilde{v}_\gamma$ sequentially from q . The target model can then evaluate every prefix

$$s, \quad s\tilde{v}_1, \quad s\tilde{v}_1\tilde{v}_2, \quad \dots, \quad s\tilde{v}_1 \cdots \tilde{v}_\gamma$$

in parallel. A causal transformer computes these next-token distributions in one forward pass on the draft block, with the usual causal mask. Sampling the draft requires sequential steps, but once its tokens are known, the target model can evaluate all their probabilities together.

At a given prefix, a proposal $v \sim q$ is accepted with probability $\min\{1, p(v)/q(v)\}$. If it is rejected, draw a replacement from the normalized positive part of $p - q$. For a real number u , write $[u]_+ = \max\{u, 0\}$.

Algorithm 1 One speculative decoding step [4, Algorithm 1]

Require: Prefix s , target p , draft model q , draft length $\gamma \geq 1$

```

1: for  $i = 1, \dots, \gamma$  do
2:    $q_i(\cdot) \leftarrow q(\cdot \mid s\tilde{v}_1 \cdots \tilde{v}_{i-1})$ 
3:   Sample  $\tilde{v}_i \sim q_i$ 
4: end for
5: In parallel, evaluate  $p_i(\cdot) \leftarrow p(\cdot \mid s\tilde{v}_1 \cdots \tilde{v}_{i-1})$ 
   for  $i = 1, \dots, \gamma + 1$ 
6: for  $i = 1, \dots, \gamma$  do
7:   Sample  $U_i \sim \text{Unif}[0, 1]$ , independently
8:   if  $U_i > \min\{1, p_i(\tilde{v}_i)/q_i(\tilde{v}_i)\}$  then
9:      $r_i(v) \leftarrow \frac{[p_i(v) - q_i(v)]_+}{\sum_{u \in \mathcal{V}} [p_i(u) - q_i(u)]_+}$  for  $v \in \mathcal{V}$ 
10:    Sample  $w \sim r_i$ 
11:    return  $s\tilde{v}_1 \cdots \tilde{v}_{i-1}w$  ▷ Discard the rest of the draft
12:   end if
13: end for
14: Sample  $w \sim p_{\gamma+1}$ 
15: return  $s\tilde{v}_1 \cdots \tilde{v}_\gamma w$  ▷ All drafts accepted; add a bonus token

```

We keep accepted proposals up to the first rejection. Later target distributions condition on the rejected token, so we discard them and start a new iteration from the corrected prefix. Repeat Algorithm 1 until an end-of-sequence token or the requested output length.

2.2 Correctness

Accepted proposals account for the probability mass shared by p and q . The replacement distribution supplies the rest of the target mass.

Lemma 2.1 (One-token speculative sampling). *For distributions p, q on $\mathcal{V}$, propose $V \sim q$ and accept it with probability $\min\{1, p(V)/q(V)\}$. On rejection, output a fresh sample from*

$$r(v) = \frac{[p(v) - q(v)]_+}{\sum_u [p(u) - q(u)]_+}.$$

The resulting output has distribution p .

Proof. The probability of accepting a proposal equal to v is

$$q(v) \min\{1, p(v)/q(v)\} = \min\{p(v), q(v)\}.$$

If $q(v) = 0$, this contribution is zero; the ratio is never evaluated on such a proposal. The overall acceptance probability is

$$\beta = \sum_v \min\{p(v), q(v)\} = 1 - \frac{1}{2} \sum_v |p(v) - q(v)| = 1 - \text{TV}(p, q), \quad (8)$$

where TV denotes total variation distance. Since $\sum_v p(v) = \sum_v q(v) = 1$, $\sum_v [p(v) - q(v)]_+ = 1 - \beta$. Consequently,

$$\begin{aligned} \mathbb{P}(\text{output} = v) &= \min\{p(v), q(v)\} + (1 - \beta)r(v) \\ &= \min\{p(v), q(v)\} + [p(v) - q(v)]_+ = p(v). \end{aligned}$$

When $\beta = 1$, $p = q$ and rejection has probability zero, so the residual distribution is never needed. This also explains why the denominator in Algorithm 1 is positive whenever its rejection branch can occur. $\square$

Proposition 2.2. *Repeated speculative decoding samples exactly from the target autoregressive distribution. In particular, for every length T and token sequence $(v_1, \dots, v_T)$ following an initial prompt s_0 , its probability is*

$$\prod_{t=1}^T p(v_t \mid s_0 v_1 \cdots v_{t-1}).$$

Proof. We may sample each draft token and acceptance uniform when we need it, since unused random draws do not affect the output. At any reached draft position, condition on the preceding output tokens and on the algorithm's previous decisions. The next proposal still has distribution q at this prefix, and its acceptance uniform is fresh. Lemma 2.1 shows that the next emitted token, whether accepted or replaced, has the target distribution p at that prefix. If the algorithm has reached a bonus-token position, it samples directly from the same target distribution instead.

In every internal state compatible with the emitted prefix, the next output therefore has conditional distribution p . Averaging over those states gives the same distribution. Starting with the first token and proceeding inductively, each output has the required conditional law. Their product gives the claimed sequence probability. $\square$

2.3 Expected progress and speedup

Although the sampler is exact for any p and q , its speed depends on the acceptance rate and the cost of running the draft model.

For a prefix s , write

$$\beta(s) = \sum_{v \in \mathcal{V}} \min\{p(v \mid s), q(v \mid s)\}.$$

This is the probability of accepting the next draft at s . Let A_i be the event that proposal i passes its acceptance test, and let $R_i = A_1 \cap \cdots \cap A_{i-1}$ be the event that position i is reached, with R_1 the certain event. Denote its draft prefix by $S_i = s\tilde{v}_1 \cdots \tilde{v}_{i-1}$.

Constant mean acceptance assumption. For the runtime calculation, assume there is a constant $\alpha \in [0, 1]$ such that, for each reachable position and each decoding iteration,

$$\mathbb{E}[\beta(S_i) \mid R_i] = \alpha. \quad (9)$$

Equivalently, $\mathbb{P}(A_i \mid R_i) = \alpha$.

Let $L \in \{0, \dots, \gamma\}$ be the number of consecutive accepted drafts and let $N = L + 1$ be the number of tokens emitted in one iteration. The extra token is either a replacement or the bonus token. By repeated conditioning and (9),

$$\mathbb{P}(L \geq j) = \mathbb{P}(A_1 \cap \dots \cap A_j) = \alpha^j, \quad 0 \leq j \leq \gamma.$$

For an integer-valued $N \in \{1, \dots, \gamma + 1\}$, the identity $N = \sum_{k=1}^{\gamma+1} \mathbf{1}\{N \geq k\}$ gives

$$\mathbb{E}[N] = \sum_{j=0}^{\gamma} \mathbb{P}(L \geq j) = \sum_{j=0}^{\gamma} \alpha^j = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}, \quad 0 \leq \alpha < 1. \quad (10)$$

Cost model. Let T_p be the wall-clock time for one ordinary target-model decoding step. Assume the target can verify the entire draft and compute the bonus-token distribution in time T_p , using sufficient parallel computation. Let T_q be the time for one draft-model step and define the cost coefficient

$$c := \frac{T_q}{T_p}.$$

Neglect acceptance-test overhead. The γ draft steps are sequential, so one speculative iteration costs

$$\gamma T_q + T_p = (1 + \gamma c) T_p. \quad (11)$$

The coefficient c depends on the model, hardware, and implementation. Verifying $\gamma + 1$ target positions in time T_p requires enough parallel computation; verification still performs arithmetic at every position.

Theorem 2.3 (Wall-clock speedup [4, Theorem 3.8]). *Under the constant mean acceptance and cost assumptions above, the speedup over ordinary target-model decoding is*

$$\text{Speedup} = \frac{\mathbb{E}[N]}{1 + \gamma c} = \frac{1 - \alpha^{\gamma+1}}{(1 - \alpha)(1 + \gamma c)}. \quad (12)$$

For $\alpha = 1$, this equals $(\gamma + 1)/(1 + \gamma c)$.

Proof. An ordinary target-model run spends time T_p per output token. Each speculative iteration spends $(1 + \gamma c)T_p$ and emits $\mathbb{E}[N]$ tokens on average. Thus the time per token is

$$\frac{(1 + \gamma c)T_p}{\mathbb{E}[N]}.$$

Dividing the baseline time T_p by this quantity proves (12). We compare total time with total progress over many complete iterations, so a final truncated iteration has negligible relative cost for long generations. Averaging the per-iteration ratios $(1 + \gamma c)T_p/N$ would give a different quantity. $\square$

A Alternative proof of lemma 1.3

We give an alternative proof of Lemma 1.3. The Hamming distance $d_H(v, w)$ counts the coordinates on which v and w differ. As in Section 1.3, $n \geq 2$ and logarithms have base two.

Lemma A.1. *If U is uniform on a nonempty set $S \subseteq [n]^n$, then for any $z \in [n]^n$,*

$$\log |S| \leq 3 \log n \mathbb{E}[d_H(U, z)].$$

Proof. Set $s = |S|$ and $D = \mathbb{E}[d_H(U, z)]$. If $s = 1$, the assertion follows from $D \geq 0$. If $s \geq 2$, at most one element of S equals z , so $D \geq (s - 1)/s$. For $s = 2, 3$, this gives $\log s \leq 3(s - 1)/s \leq 3D \log n$.

Now suppose $s \geq 4$, so $D \geq 3/4$. Let $e_j = \mathbb{P}(U_j \neq z_j)$. To describe U_j , first specify whether it equals z_j , then, if it does not, specify one of at most $n - 1$ values. The entropy chain rule therefore gives

$$H(U_j) \leq h_2(e_j) + e_j \log(n - 1).$$

Entropy subadditivity, concavity of h_2 , and $\sum_j e_j = D$ yield

$$\begin{aligned} \log s = H(U) &\leq \sum_{j=1}^n H(U_j) \\ &\leq n h_2(D/n) + D \log(n - 1) \\ &\leq D \log\left(\frac{en(n - 1)}{D}\right) \leq 3D \log n. \end{aligned}$$

Here e is Euler's number. The penultimate inequality uses $h_2(u) \leq u \log(e/u)$, which follows from $-(1 - u) \ln(1 - u) \leq u$. For the last inequality, $D \geq 3/4$ and $n^2 \geq 4(n - 1)$ imply $en(n - 1)/D \leq (4e/3)n(n - 1) \leq n^3$, since $e < 3$. $\square$

Alternative proof of Lemma 1.3. Let $F = (f(1), \dots, f(n))$ and let M denote the message. Conditional on $M = m$, the vector F is uniform on the set S_m of tables producing m . Let $z_m(j)$ be a most likely value of F_j under this conditional distribution, with ties broken arbitrarily. This minimizes the error for each requested index. Thus the minimum conditional error, averaged over the uniform J , is

$$\delta_m = \frac{1}{n} \mathbb{E}[d_H(F, z_m) \mid M = m].$$

By Lemma A.1, $H(F \mid M = m) = \log |S_m| \leq 3n \log n \delta_m$. On the other hand, the entropy chain rule and the fact that M is a function of F give

$$H(F \mid M) = H(F) - H(M) \geq n \log n - B.$$

Averaging over m proves the claimed error bound. Additional information independent of F conditional on J does not change the conditional distribution of F given (M, J) , so it cannot improve on the coordinate-wise most likely answers. Randomized protocols satisfy the same bound by fixing their random coins, applying the argument, and then averaging. $\square$

References

- [1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 2022. <https://arxiv.org/abs/2201.11903>.
- [2] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. *Advances in Neural Information Processing Systems*, 2022. <https://arxiv.org/abs/2203.14465>.
- [3] Binghui Peng, Srini Narayanan, and Christos Papadimitriou. On Limitations of the Transformer Architecture. 2024. <https://arxiv.org/abs/2402.08164>.
- [4] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. *International Conference on Machine Learning*, 2023. <https://arxiv.org/abs/2211.17192>.

AI Use. I prepared these lecture notes with the aid of AI assistance; I take full responsibility for all content in the notes.