Fast Inference from Transformers via Speculative Decoding

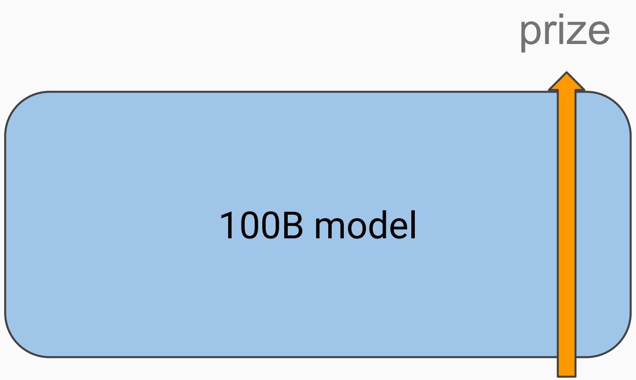
Dongwei Jiang, Yiran Zhong, Oct 22

Highlevel TakeAway

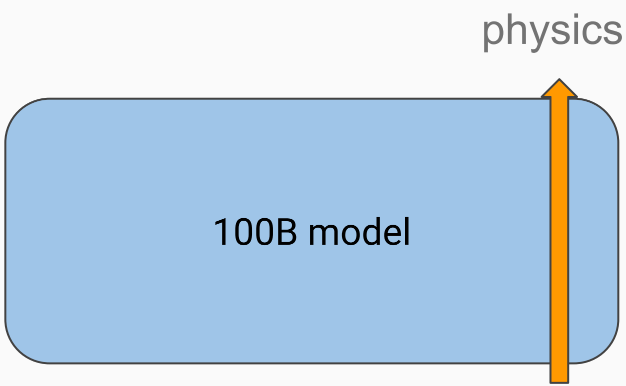
- This is a method that decodes faster from autoregressive models: **2X-3X** in typical scenarios.
- Only different decoding algorithm: **no architecture changes, no re-training**.
- **Identical** output distribution.

Observation

Some tokens are easier to predict than others. So it's possible for efficient smaller models to stand in for their larger counterparts!



Geoffrey Hinton was awarded the nobel …



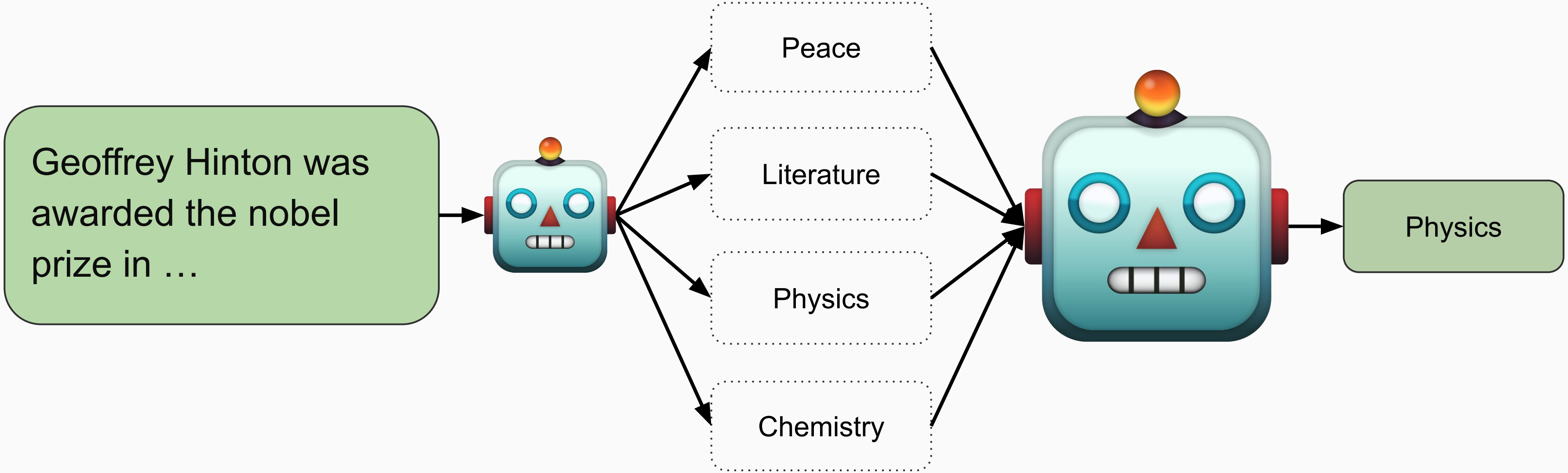
Geoffrey Hinton was awarded the nobel prize in …

This is easy!
Let's use 1B model!

This is hard!
Have to use 100B model!

How can we make sure small model's predictions are correct?

We can use small model to propose several candidates and use large model to decide whether to accept them!



Since large models can verify multiple predictions at once, we can let the small model guess several tokens ahead!

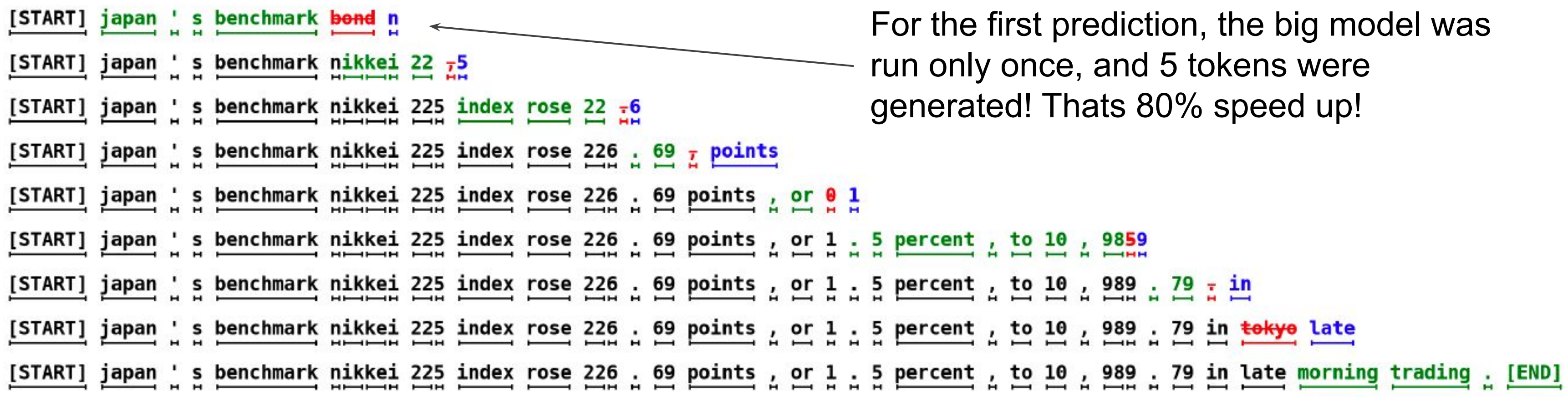
Speculative Decoding

Let the small model make multiple expensive step-by-step predictions, then verify them all at once with the big model!

Green ones are predictions from the small model

Red ones are rejected predictions from the big model

Blue ones are corrections proposed by the big model



For the first prediction, the big model was run only once, and 5 tokens were generated! Thats 80% speed up!

Overall Algorithm

Definition: M_p is the base model and M_q is the efficient approximation model

To sample $x \sim p(x)$, we instead sample $x \sim q(x)$

$q(x) = q(x_n|x_{<n})$

Random numbers sampled uniformly from [0,1]

Find the min i that satisfies the condition. If $p_i(x)$ > $q_i(x)$, it's always accepted. If everything is accepted, it sets n to γ

Deal with the last token: if everything is accepted, it follows the distribution of M_p

Else, follows the distribution of norm(max(0, $p_{n+1}(x)$ - $q_{n+1}(x)$)). This adjusts the probability mass that was affected by the rejected guess by M_q from step n + 1

Algorithm 1 SpeculativeDecodingStep

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

We're substituting multiple AR generation of M_p with multiple AR generation of M_q and one parallel generation of M_p

What is the Expected Number of Accepted Tokens?

Definition: $\beta_{x_{<t}}$ is the probability of accepting $x_t \sim q(x_t|x_{<t})$, under i.i.d assumption, $\alpha = E(\beta)$ is its expectation, or the probability of accepting any token, then we have:

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

First, get the probability mass function of # generated tokens (or X).

For k = 1, 2, ..., γ : $P(X=k) = (1-\alpha)\alpha^{k-1}$

For k = γ + 1: $P(X=k) = \alpha^{\gamma}$

Then, the expected value of X is $E(X) = (1-\alpha)\sum_{k=1}^{\gamma} k\alpha^{k-1} + (\gamma+1)\alpha^{\gamma}$

Lemma: the sum of a geometric series (proof by induction): $\sum_{k=1}^{n} kx^{k-1} = \frac{1-x^n}{(1-x)^2} - \frac{nx^n}{1-x}$

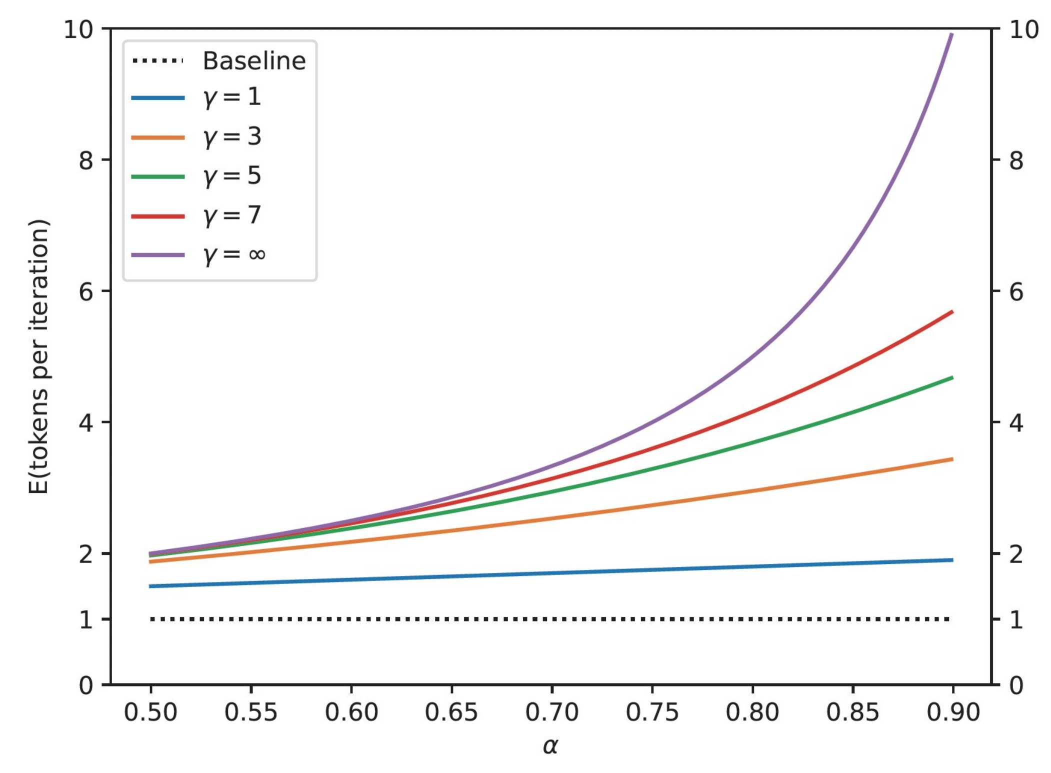
Applying this formula above with x = α and n = γ : $E(X) = (1-\alpha)\left[\frac{1-\alpha^{\gamma}}{(1-\alpha)^2} - \frac{\gamma\alpha^{\gamma}}{1-\alpha}\right] + (\gamma+1)\alpha^{\gamma}$

Empirical results for Expected Number of Accepted Tokens

Remember:

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

If α is 0.9, we can almost accept all γ ! But what are the true alphas?



Calculating Alpha

Define: $D_{LK}(p,q) = \sum_x |p(x) - M(x)| = \sum_x |q(x) - M(x)|$ where $M(x) = \frac{p(x)+q(x)}{2}$

Lemma: $D_{LK}(p,q) = 1 - \sum_x \min(p(x), q(x))$

Proof: $D_{LK}(p,q) = \sum_x |p(x) - M(x)| = \sum_x \frac{|p-q|}{2} = 1 - \sum_x \frac{p+q-|p-q|}{2} = 1 - \sum_x \min(p(x), q(x))$

We also have: $\beta = 1 - D_{LK}(p,q)$

Proof: $\beta = \mathbb{E}_{x\sim q(x)} \begin{cases} 1 & \text{if } q(x) \leq p(x) \\ \frac{p(x)}{q(x)} & \text{if } q(x) > p(x) \end{cases} = E_{x\sim q(x)} \min(1, \frac{p(x)}{q(x)}) = \sum_x \min(p(x), q(x))$

So we also have: $\alpha = 1 - E(D_{LK}(p,q)) = E(\min(p,q))$

So we only need look at how much overlap there is between the distributions p and q!

Walltime improvement

To analyze improvement of the actual elapsed time for running algorithm of speculative decoding

Reduction in calls: $\frac{1-\alpha^{\gamma+1}}{1-\alpha}$ (reduce the # of call to target model M_p)

Cost efficient: c (ratio of time for single run of approximation model M_q to target model M_p

Expected cost producing a token: $\frac{(c\gamma+1)(1-\alpha)}{1-\alpha^{\gamma+1}} \cdot T$ where T is cost of single decoding step.

Improvement Factor: $\frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$ the higher value of alpha and lower c for better improvement

Special case when $\gamma = 1$

When $\alpha > c$, there's value of γ provide improvement

When $\gamma = 1$, The improvement factor is $\dfrac{1-\alpha^2}{(1-\alpha)(c+1)} = \dfrac{1+\alpha}{1+c}$

Number of Arithmetic operations

Purpose: Analyze how speculative decoding impacts the total number of arithmetic operations compared to standard decoding.

Speculative decoding involve $\gamma + 1$ parallel runs of M_p it increase the number of concurrent arithmetic operations by a factor of $\gamma + 1$.

Increase concurrency will cause the unnecessary additional computation if samples were rejected.

Expected factor of increase

Suppose single run of M_p has $\widehat{T}$ operations, and M_q has $c \cdot \widehat{T}$ operations

The γ run for M_q and the $\gamma + 1$ run in parallel for M_p

Total operation: $\widehat{T}c\gamma + \widehat{T}(\gamma + 1)$

We normalize (show total operations of speculative decoding compare to the standard decoding) it by $\widehat{T}$ and dividing by expected number of tokens.

The expected factor of increase in operations: $\dfrac{(1-\alpha)(\gamma c + \gamma + 1)}{1 - \alpha^{\gamma+1}}$

Memory Efficiency

Memory shrink by factor $\frac{1-\alpha^{\gamma+1}}{1-\alpha}$ (expected token generated)

Because comparing to standard decoding which generating one token at time, we generate this expected number of token in parallel, the target model's weights and KV cache can be read once per execution.

Choice of γ

γ represents the number of speculative decoding iterations before a target model evaluation

goal is to maximize walltime improvement by selecting an optimal γ given cost coefficient c and acceptance rate α

γ Could be optimized through numerical search.

The best γ depends on balance between computation cost c and acceptance rate α

Trade off (γ increase with higher α and lower c)

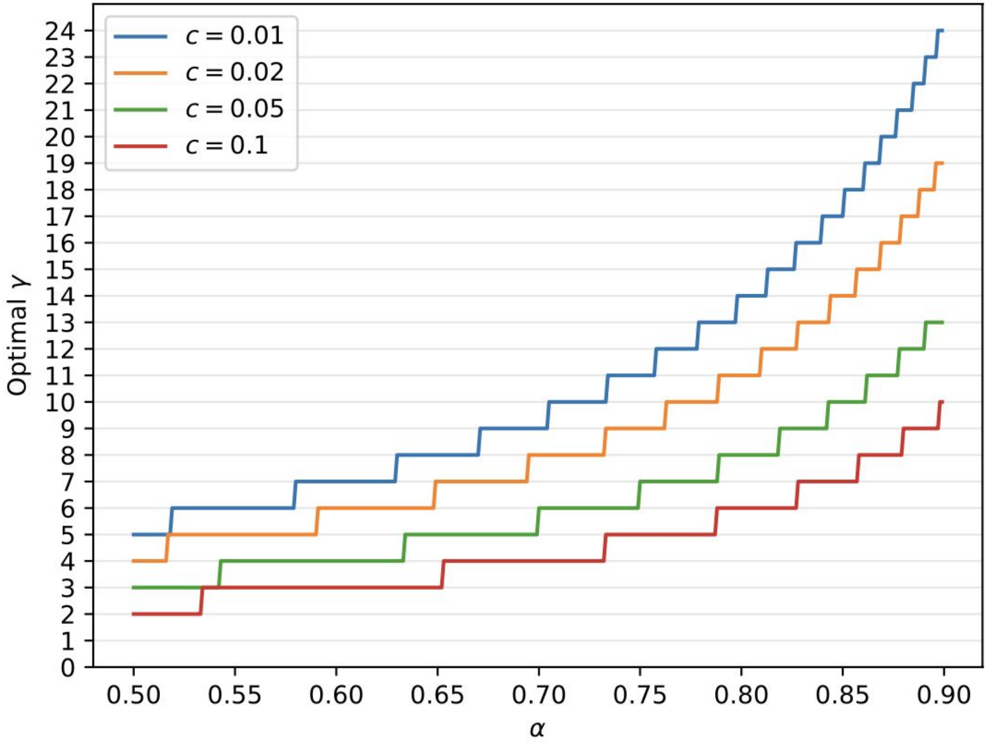


Figure 3. The optimal γ as a function of α for various values of c.

Increase of α lead to speed up and lower increase in arithmetic operation

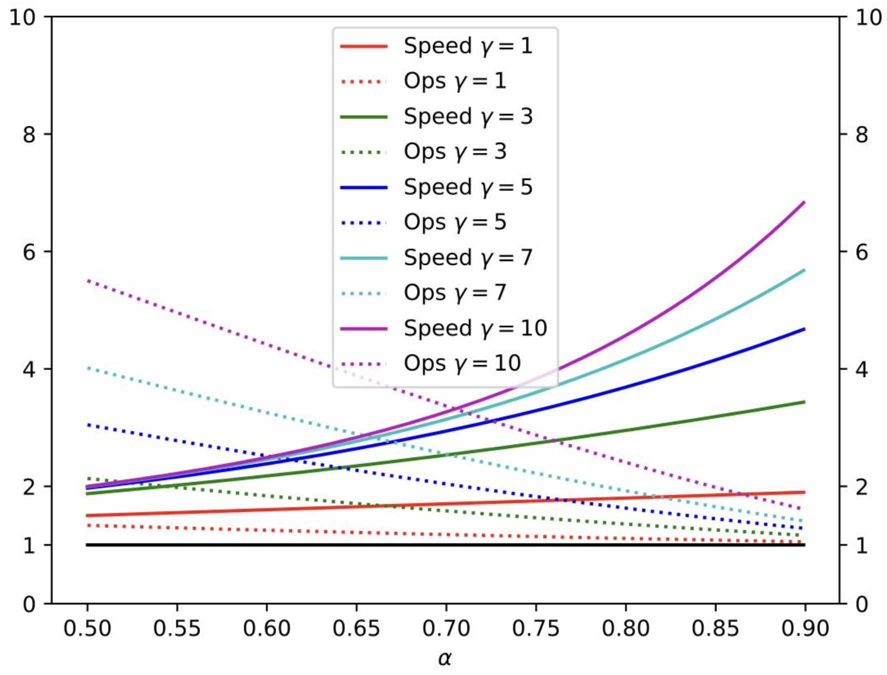


Figure 4. The speedup factor and the increase in number of arithmetic operations as a function of α for various values of γ.

Different γ affect speed and operations

Table 1. The total number of arithmetic operations and the inference speed vs the baseline, for various values of γ and α, assuming $c = \hat{c} = 0$.

α	γ	OPERATIONS	SPEED
0.6	2	1.53X	1.96X
0.7	3	1.58X	2.53X
0.8	2	1.23X	2.44X
0.8	5	1.63X	3.69X
0.9	2	1.11X	2.71X
0.9	10	1.60X	6.86X

Observation 4

Adaptive speculative decoding offers a more efficient way to boost speed by intelligently managing the number of speculative steps, leading to significant walltime reductions.

However, if γ is too high relative to acceptance rate α, it may lead to wasted computation due to rejected tokens from M_q, thus reduce efficiency.

Speculative decoding with different γ compared to standard decoding

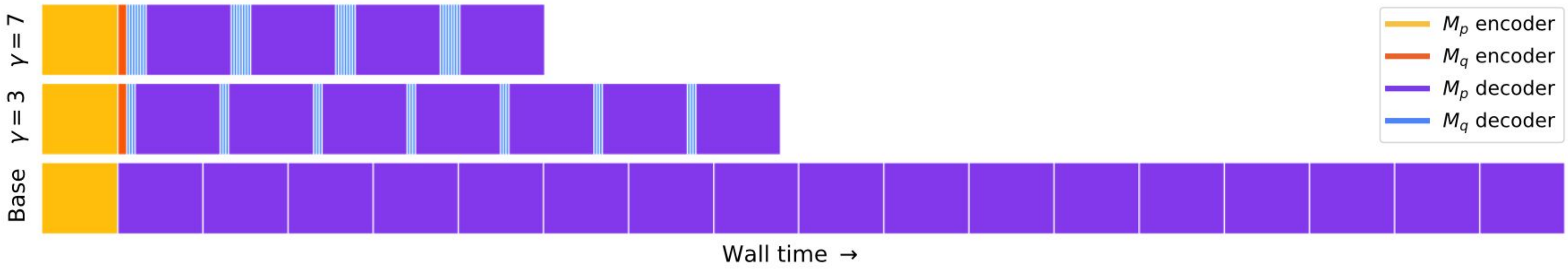


Figure 5. A simplified trace diagram for a full encoder-decoder Transformer stack. The top row shows speculative decoding with $\gamma = 7$ so each of the calls to M_p (the purple blocks) is preceded by 7 calls to M_q (the blue blocks). The yellow block on the left is the call to the encoder for M_p and the orange block is the call to the encoder for M_q. Likewise the middle row shows speculative decoding with $\gamma = 3$, and the bottom row shows standard decoding.

Experiment

Goal: Validate the speculative decoding method on real tasks and compare it with standard decoding.

Task: 1. Machine Translation (English-to-German) 2. Text Summarization

Models:

Target Model: T5-XXL (11B parameters)

Approximation Models: T5-small (77M), T5-base (250M), T5-large (800M)

Decoding type: Argmax Sampling (temperature = 0)

Standard Sampling (temperature = 1)

Result

Translation Task: WMT EnDe fine-tuned on T5. **Summarization Task**: CNN/DM fine-tuned on T5.

Table 2. Empirical results for speeding up inference from a T5-XXL 11B model.

TASK	M_q	TEMP	γ	α	SPEED
ENDE	T5-SMALL ★	0	7	0.75	**3.4X**
ENDE	T5-BASE	0	7	0.8	2.8X
ENDE	T5-LARGE	0	7	0.82	1.7X
ENDE	T5-SMALL ★	1	7	0.62	**2.6X**
ENDE	T5-BASE	1	5	0.68	2.4X
ENDE	T5-LARGE	1	3	0.71	1.4X
CNNDM	T5-SMALL ★	0	5	0.65	**3.1X**
CNNDM	T5-BASE	0	5	0.73	3.0X
CNNDM	T5-LARGE	0	3	0.74	2.2X
CNNDM	T5-SMALL ★	1	5	0.53	**2.3X**
CNNDM	T5-BASE	1	3	0.55	2.2X
CNNDM	T5-LARGE	1	3	0.56	1.7X

Observation 5

Argmax sampling provide higher acceptance rate α and thus have better speed up improvement.

Approximation model T5-small achieve best improve in speed

Empirical α value for various target model

Smaller M_q (like unigrams/bigrams) lead to lower α

Simple models (unigrams, bigrams) still provide non-zero α , may yielding reasonable speedups.

M_p	M_q	SMPL	α
GPT-LIKE (97M)	UNIGRAM	T=0	0.03
GPT-LIKE (97M)	BIGRAM	T=0	0.05
GPT-LIKE (97M)	GPT-LIKE (6M)	T=0	0.88
GPT-LIKE (97M)	UNIGRAM	T=1	0.03
GPT-LIKE (97M)	BIGRAM	T=1	0.05
GPT-LIKE (97M)	GPT-LIKE (6M)	T=1	0.89
T5-XXL (ENDE)	UNIGRAM	T=0	0.08
T5-XXL (ENDE)	BIGRAM	T=0	0.20
T5-XXL (ENDE)	T5-SMALL	T=0	0.75
T5-XXL (ENDE)	T5-BASE	T=0	0.80
T5-XXL (ENDE)	T5-LARGE	T=0	0.82
T5-XXL (ENDE)	UNIGRAM	T=1	0.07
T5-XXL (ENDE)	BIGRAM	T=1	0.19
T5-XXL (ENDE)	T5-SMALL	T=1	0.62
T5-XXL (ENDE)	T5-BASE	T=1	0.68
T5-XXL (ENDE)	T5-LARGE	T=1	0.71

T5-XXL (CNNDM)	UNIGRAM	T=0	0.13
T5-XXL (CNNDM)	BIGRAM	T=0	0.23
T5-XXL (CNNDM)	T5-SMALL	T=0	0.65
T5-XXL (CNNDM)	T5-BASE	T=0	0.73
T5-XXL (CNNDM)	T5-LARGE	T=0	0.74
T5-XXL (CNNDM)	UNIGRAM	T=1	0.08
T5-XXL (CNNDM)	BIGRAM	T=1	0.16
T5-XXL (CNNDM)	T5-SMALL	T=1	0.53
T5-XXL (CNNDM)	T5-BASE	T=1	0.55
T5-XXL (CNNDM)	T5-LARGE	T=1	0.56
LAMDA (137B)	LAMDA (100M)	T=0	0.61
LAMDA (137B)	LAMDA (2B)	T=0	0.71
LAMDA (137B)	LAMDA (8B)	T=0	0.75
LAMDA (137B)	LAMDA (100M)	T=1	0.57
LAMDA (137B)	LAMDA (2B)	T=1	0.71
LAMDA (137B)	LAMDA (8B)	T=1	0.74

Thank you!