Speculative decoding

Polina Shaydurova, 212

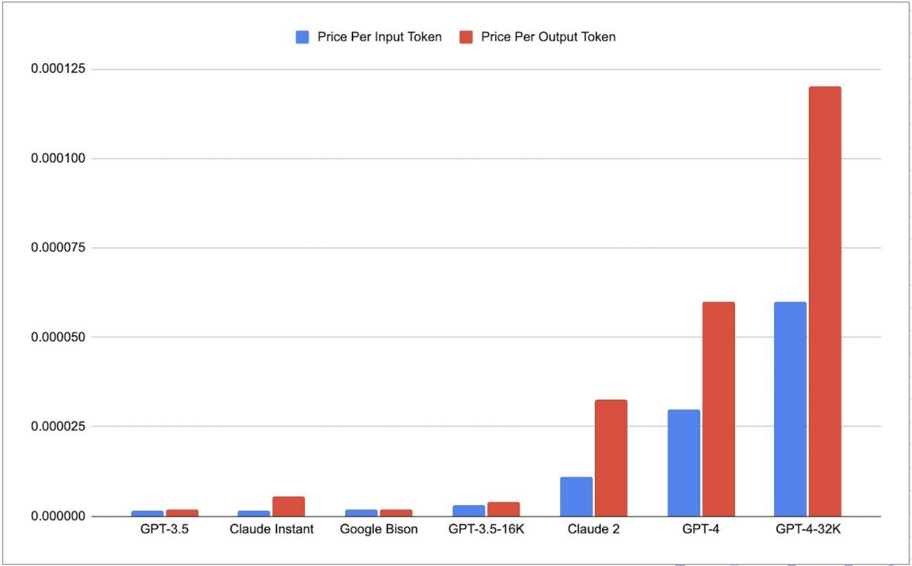
April 1, 2024

Introduction

Inference problems

- The inability to make the inference parallel
- The main latency bottleneck arising from memory reads/writes rather than arithmetic computations
- Increasing the batch size introduces higher latency

Inference Cost

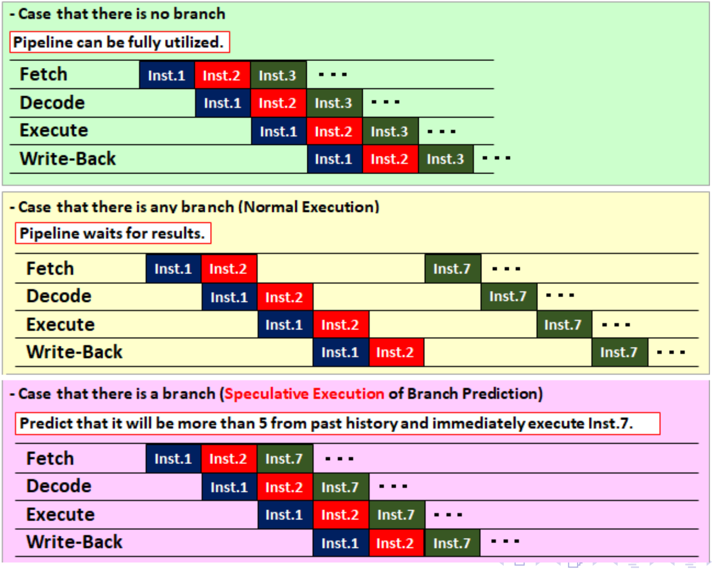


Early solutions

- Distillation, sparcification, quantization, architecture modification
- Adaptive computation methods
- Blockwise Parallel Decoding
- Shallow Aggressive Decoding

Speculative decoding

Speculative execution



Overview

- M_p — large model, $p(x|x_{<t})$ — distribution of output with prefix $x_{<t}$
- M_q — more efficient approximation model, $q(x|x_{<t})$ — distribution of output with prefix $x_{<t}$
- Want to use M_q for generating γ tokens with evaluating all guesses with M_p

Speculative sampling. Idea

1. $x \sim q(x)$
2. If $q(x) \leq p(x)$ then accept this token
3. Else reject this token with $P = 1 - \frac{p(x)}{q(x)}$.
4. If token was rejected, generating it again from distribution $p'(w) = \text{norm}(\max(0, p(w) - q(w))$

Speculative sampling. Algorithm

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 $prefix + [x_1, \ldots, x_n, t]$

Speculative sampling. Example

[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~,~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or ~~0~~ 1

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 9859

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 ~~,~~ in

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in ~~tokyo~~ late

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]

Figure 1. Our technique illustrated in the case of unconditional language modeling. Each line represents one iteration of the algorithm. The **green** tokens are the suggestions made by the approximation model (here, a GPT-like Transformer decoder with 6M parameters trained on lm1b with 8k tokens) that the target model (here, a GPT-like Transformer decoder with 97M parameters in the same setting) accepted, while the **red** and **blue** tokens are the rejected suggestions and their corrections, respectively. For example, in the first line the target model was run only once, and 5 tokens were generated.

Number of generating tokens

- β — **acceptance rate**, probability to accept $x_t \sim q(x_t|x_{<t})$
- $\alpha = \mathbb{E}[\beta]$
- $\alpha = \mathbb{E}[\min(p, q)]$
- $\#$ generated tokens $\sim \max(\text{Geom}(1-\alpha), 1+\gamma)$
- $\mathbb{E}[\#\text{ generated tokens}] = \frac{1-\alpha^{\gamma+1}}{1-\alpha}$

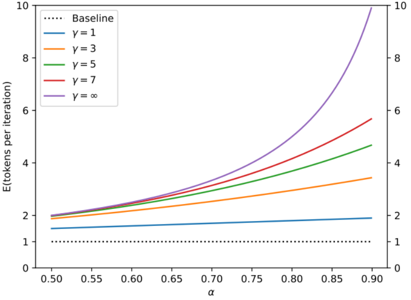


Figure 2. The expected number of tokens generated by Algorithm 1 as a function of α for various values of γ.

Walltime improvement

- c — **cost coefficient** equals to $\frac{\text{time for a single run of } M_q}{\text{time for a single run of } M_p}$
- $\mathbb{E}[\text{improvement factor in total walltime}] = \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$

Number of arithmetic operations

- $\hat{c} = \frac{\text{arithmetic operations per token of} M_q}{\text{arithmetic operations per token of} M_p}$
- $\mathbb{E}[\text{factor of increase in the number of operations}] = \frac{(1-\alpha)(\gamma\hat{c}+\gamma+1)}{1-\alpha^{\gamma+1}}$
- The number of memory accesses for reading them shrinks by a factor of $\frac{1-\alpha^{\gamma+1}}{1-\alpha}$

Choosing γ

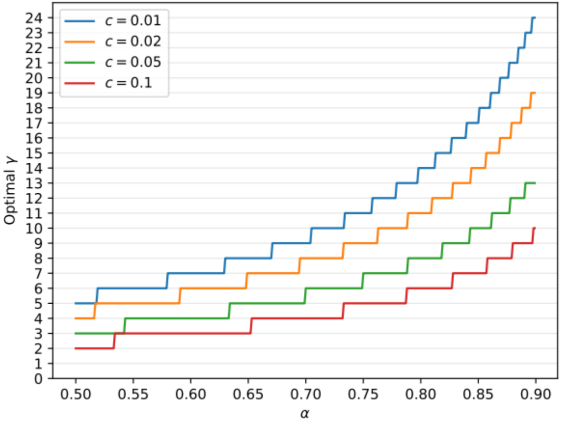


Figure 3. The optimal γ as a function of α for various values of c.

Choosing γ

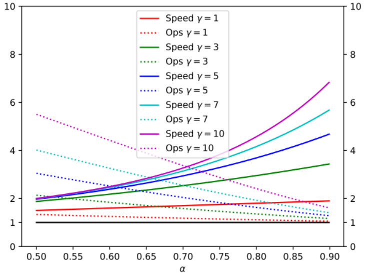


Figure 4. The speedup factor and the increase in number of arithmetic operations as a function of α for various values of γ.

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

Choosing γ

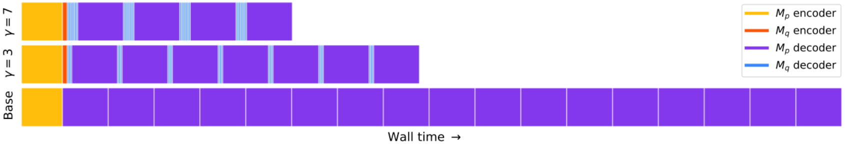


Figure 5. A simplified trace diagram for a full encoder-decoder Transformer stack. The top row shows speculative decoding with $\gamma = 7$ so each of the calls to M_p (the purple blocks) is preceded by 7 calls to M_q (the blue blocks). The yellow block on the left is the call to the encoder for M_p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

Approximation models

- Off-the-shelf smaller Transformers
- Negligible-cost models, n-gram models
- Non-autoregressive models
- Chooses tokens at random

Experiments

Empirical results for speeding up inference

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

Empirical α values

Table 3. Empirical α values for various target models M_p, approximation models M_q, and sampling settings. T=0 and T=1 denote argmax and standard sampling respectively[6].

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

Speculative decoding conclusions

Conclusions

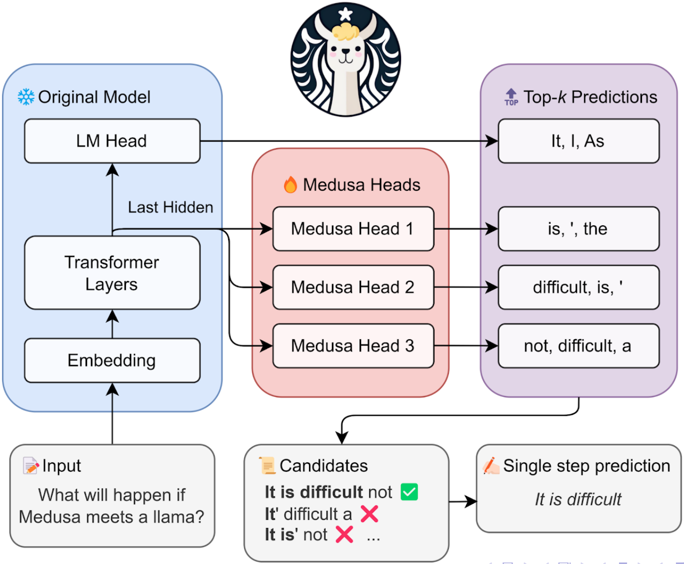
- No changing of the model architecture
- No retraining
- The same output distribution
- Can use out-of-the-box models

Medusa

Is speculative decoding the ultimate solution?

- Finding the ideal approximation model
- System complexity
- Need additional computation resources

Main idea


Original Model
LM Head
Last Hidden
Transformer Layers
Embedding
Medusa Heads
Medusa Head 1
Medusa Head 2
Medusa Head 3
Top-*k* Predictions
It, I, As
is, ', the
difficult, is, '
not, difficult, a
Input
What will happen if Medusa meets a llama?
Candidates
It is difficult not
It' difficult a
It is' not ...
Single step prediction
It is difficult

Medusa heads

- head is a single layer of feed-forward network with a residual connection for each head
- The original model remains static; only the Medusa heads are fine-tuned
- A top-1 accuracy rate of approximately 60% for predicting the 'next-next' token

Tree attention

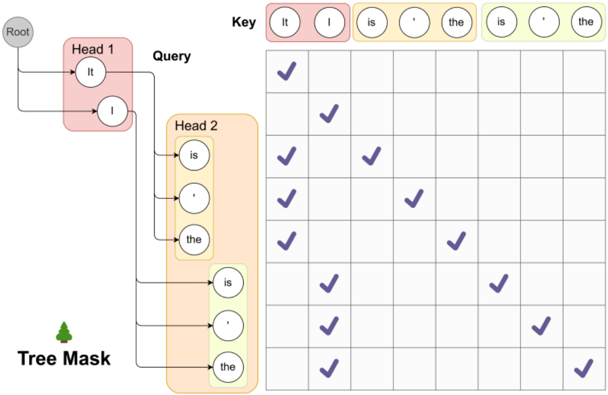


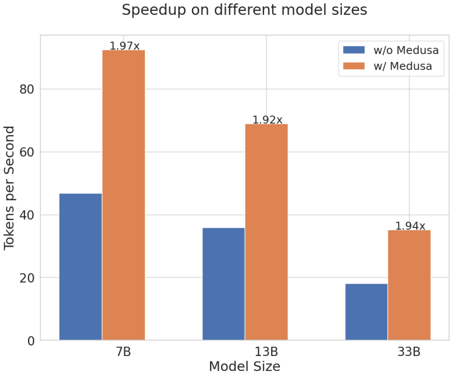
Figure 2: **Tree Attention Illustrated.** This visualization demonstrates the use of tree attention to process multiple candidates concurrently. As exemplified, the top-2 predictions from the first MEDUSA head and the top-3 from the second result in a total of $2 \times 3 = 6$ candidates. Each of these candidates corresponds to a distinct branch within the tree structure. To guarantee that each token only accesses its predecessors, we devise an attention mask that exclusively permits attention flow from the current token back to its antecedent tokens. The positional indices for positional encoding are adjusted in line with this structure.

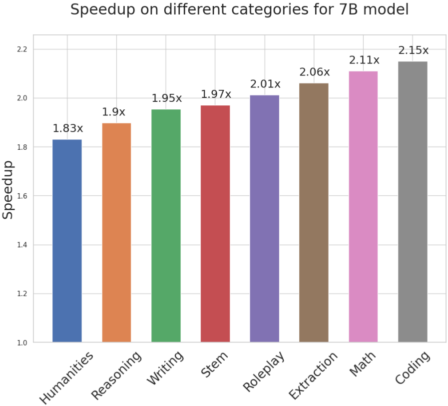
Typical acceptance

$$p_{\text{original}}(x_{n+k}|x_1, x_2, \cdots, x_{n+k-1}) > \min\left(\epsilon, \delta \exp\left(-H(p_{\text{original}}(\cdot|x_1, x_2, \cdots, x_{n+k-1}))\right)\right)$$

Medusa experiments

Experiments





Medusa conclusions

Conclusions

- No separate model
- Simple integration to existing systems
- Sampling temperature