Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan, Matan Kalman, Yossi Matias

ICML 2023

Presenter: Lingzhi Zhao

Some slides are from the authors'

Motivation

- Decoding K tokens takes K **serial runs**

- Can we somehow decode several tokens in parallel?

Previous Approaches

- Reduce the inference cost for all inputs equally
  - Distillation (Hinton, 2015), sparsification (Jaszczur, 2021), quantization (Hubara, 2016)
- Adaptive computation
  - Han, 2021, Sukhbaatar, 2019
  - Different inference steps require different size of model

They require changing the model architecture, training procedure, and re-training the models without maintaining identical outputs

Observation 1

- Some tokens are easier than others
- Hebrew: .המבוא קרב היה אישנה English: The president was Barack Obama.

Easy - e.g. can guess based on just the last token.

Hard - e.g. requires looking several tokens back, knowledge of hebrew

Observation 2

- Decoding from large transformers is memory bound

Hardware can do	Transformers need
XXX Floating point operations per byte read	**X** Floating point operations per byte read

Speculative Decoding

- Sample generations from more efficient *approximation* models as speculative prefixes for the slower *target* models
- Consider two models M_q, target model and M_p, more efficient approximation model

$$p_1(x) = M_p(pf) \longrightarrow x_1$$

$$p_2(x) = M_p(pf, x_1) \longrightarrow x_2$$

…

$$p_5(x) = M_p(pf, x_1, x_2, x_3, x_4) \longrightarrow x_5$$

Run approximation model γ steps

Speculative Decoding

- Consider two models M_q, target model and M_p, more efficient approximation model

$$q_1(x), q_2(x), q_3(x), q_4(x), q_5(x), q_6(x)$$

$$= M_q(pf, x_1, x_2, x_3, x_4, x_5)$$

Run target model once

Speculative Decoding

- Case 1: if $q(x) \geq p(x)$, then accept the generated token from the approximation model

- Case 2: if $q(x) < p(x)$, then accept with probability $\frac{q(x)}{p(x)}$
  - If rejected, sample x from an adjusted distribution $(q(x) - p(x))_+$

Theoretical Analysis: Number of Parallel Tokens

- The expected number of tokens generated by speculative decoding is
- $E(\#\ generated\ tokens) = \frac{1 - \alpha^{\gamma+1}}{1-\alpha}$
- α:expected acceptance rate
- Optimally choose the number of tokens γ to attempt to parallelize

Theoretical Analysis: Walltime Improvement

- The expected improvement factor in total walltime:

$$\frac{1 - \alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$$

- c:the ratio between the time for a single run of the approximation model and the time for a single run of the target model

How to choose γ

- The optimal γ should maximize the walltime reduction

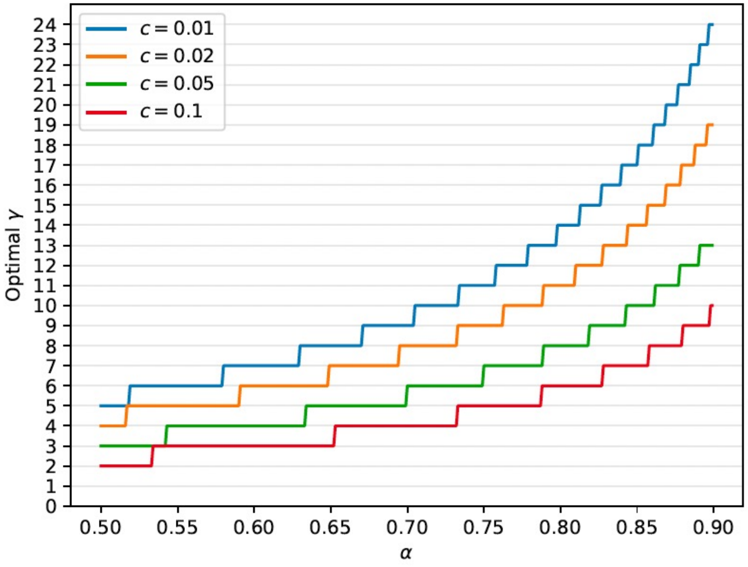

Figure 3. The optimal γ as a function of α for various values of c.

Evaluation

- Implement SD in T5X codebase; two tasks: translation and text summarization;
- Target model (11B); approximation models (800M, 250M, 77M)
- Batch size 1 on a single TPU-v4

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

- T5-small (77M) has a good balance between **acceptance rate** and number of **generated tokens,** and achieves fast inference time

Evaluation

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

- Approximation tends to produce α between 0.5 and 0.9
- Even trivial unigram and bigram approximations yield non negligible α values with negligible runtime

What SD is good at

- Decode faster from autoregressive models: 2x-3x in typical scenarios
- Only different decoding algorithm: no architecture changes, no re-training
- Identical output distribution

What SD is limited at

- Adaptively choosing γ during runtime could further improve its performance
- Fine-tune the approximation model to generate more similar distributions with the target model
- Lack comparisons with state-of-the-arts