Fast Inference from Transformers via Speculative Decoding (ICML 2023)

Akira Kawata

Abstract

- Problem
  - We need to run LLM model K times to get K tokens.
- Proposed method
  - Speculative decode
  - This enables us to run multiple token decode in parallel.
  - This can be applied to any model without any modification.
- Result
  - They achieved 2-3 times speed up.
  - After applied this method, the result token distribution exactly matches with the original one.

2. Speculative Decoding

2.1. Overview

- 構成要素
  - ターゲットモデル (Mp)：高速化したい大型モデル（高品質だが遅い）
  - 近似モデル (Mq)：下書き用の小型モデル（低コストで高速）
- プロセス
  - 下書き (Drafting)：小型モデル Mq が γ 個のトークンを先読み生成
  - 並列検証 (Parallel Evaluation)：大型モデル Mp が、それらの候補を一度の実行で並列に検証
  - 確定と修正 (Correction)：採用されたトークンを確定。最初の不採用箇所を Mp が修正し、新たなトークンとして出力
- メリット
  - 効率性：1回の重い計算（Mp）で、最大 γ+1 個のトークンを生成可能
  - 品質保証：出力される分布は、大型モデル単体の場合と完全に同一

2.2. Standardized Sampling

- argmax、top-k、nucleus、温度（temperature）調整など、一般に異なる処理として扱われる手法を、単一の共通フレームワークで扱う。
- あらゆるサンプリング手法は、最終的に調整された確率分布からの標準的なサンプリングとして定義できる。
- 具体例（argmax）：最大値以外の確率をゼロにして正規化した分布からのサンプリング、と読み替えることが可能。
- 本論文の以降の議論では、ターゲットモデル（M p ）と近似モデル（M q）から得られる分布 p(x) および q(x) は、指定の手法ですでに調整済みであると仮定する。
- これにより、個別のサンプリングアルゴリズムに依存しない、汎用的な推論高速化の議論が可能になる

2.3 Speculative Sampling

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 can predict tokens from **Mp** using tokens from **Mq** exactly!!!

Example implementation: Speculative-Sampling/speculative_sampling.py

- ▷ Sample γ guesses x1,...,γ from Mq autoregressively.
- ▷ Run Mp in parallel.
  - Input: [n_batch, γ] / int32
  - output:[n_batch, γ, n_vocab] / float32
- ▷ Determine the number of accepted guesses n.
  - Control flow changes depending on the calculation.
- ▷ Adjust the distribution from Mp if needed.
- Sample from the adjusted distribution
- ▷ Return one token from Mp, and n tokens from Mq.

あとでじっくり見たほうがいいかも

3. Analysis

3.1. Number of Generated Tokens

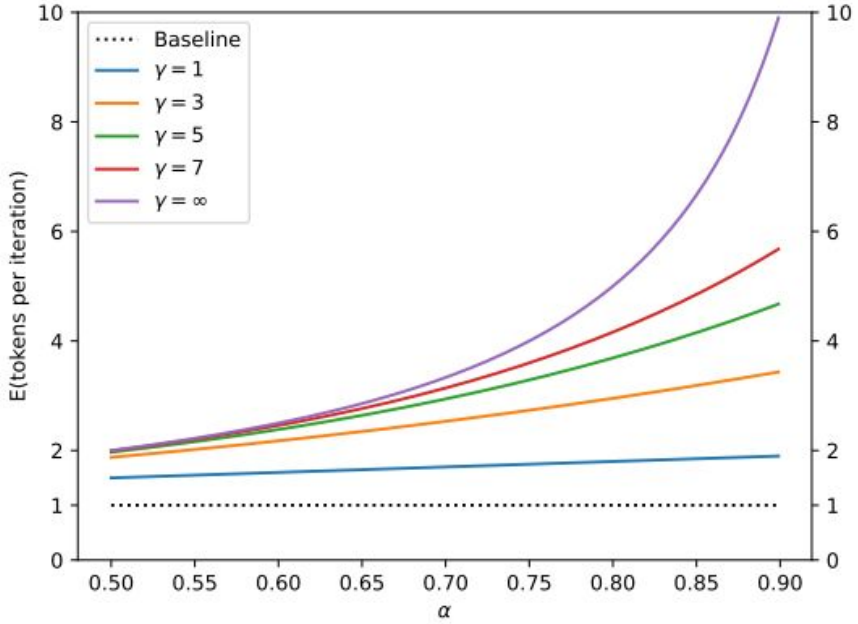


Figure 2. The expected number of tokens generated by Algorithm 1 as a function of α for various values of γ.

- Vertical axis: The number of tokens generated per single speculative decoding step
- γ: The number of tokens generated by the draft model.
- α: The degree of alignment between the probability distributions of Mq and Mp
- So, red line should be close to 7 at α = 1.0.

3.2 Calculating α

飛ばす

3.3. Walltime Improvement

- α：近似モデルが生成したトークンをターゲットモデルが採用する平均的な確率 (1.0 に近いほうがよい)
- c：ターゲットモデルの 1 回の実行時間に対する、近似モデルの実行時間の比率 (小さいほうがよい)
- T: Speculative decode なしの 1 token の生成時間
- γ: Speculative に生成するトークン数 (αとcに依存して最適値が決定)
- Speculative decode ありでの 1 token の生成時間は以下の式になる

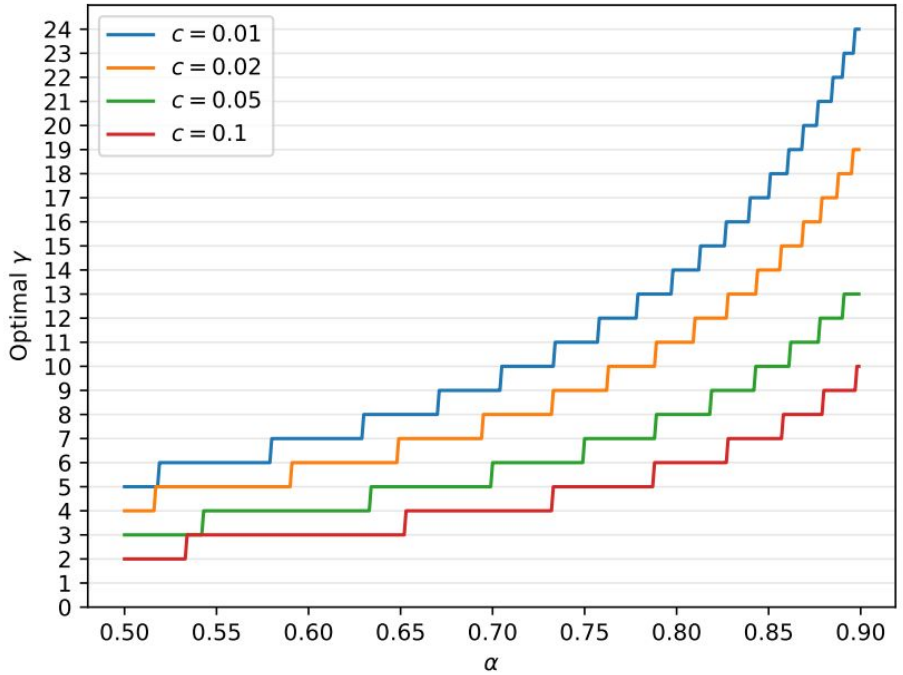
$$\frac{(c\gamma+1)(1-\alpha)}{1-\alpha^{\gamma+1}}T$$

3.4. Number of Arithmetic Operations

- 演算量は以下の式に従って増える
  - Mp の並列実行部分が無駄になるケースがあるので
- c^ は近似モデルとターゲットモデルの演算量の比
- 例えば α = 0 (ドラフトモデルがまったく当たらない)、c^ = 0 (ドラフトモデルの計算量が0) を入れると、計算量は γ + 1 倍になる

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

3.5. Choosing γ (= どれだけ予測すればよいのか)



- ドラフトモデルの性能がよいほど (= αが高い)、ドラフトモデルの計算コストが安いほど (c が小さい)、たくさん予測した法がよい

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

3.6. Approximation Models

- Speculative decode はどんなドラフトモデルを使ってもよい!
- ドラフトモデルの候補
  - 二桁小さい Transformer モデル
  - n-gram モデル (テーブル引き)
  - コンテキストからトークンをコピーするモデル
  - 非自己回帰モデル
  - ランダム

4. Experiments

4.1. Empirical Walltime Improvement

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

- ドラフトモデルとしては1/100ぐらいのモデルがよい
  - T5-XXL (11B)
  - T5-Large (800M)
  - T5-Base (250M)
  - T5-Small (77M)

4.2. 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

- 様々なタスク、サンプリング手法、モデル構成において、理論上の期待採択率 (α) が実際にどの程度の値になるか調査した
- 超大型モデルにおいても、1/100 ぐらいのモデルで高い採択率を維持できる

5. Related work

5. Related work

SKIP!

6. Discussion

6. Discussion

SKIP!