

Speculative Decoding

- [Fast Inference from Transformers via Speculative Decoding](#) (Google, 2022.11)
- [Accelerating Large Language Model Decoding with Speculative Sampling](#) (Deepmind 2023.02)

미리보기👁👁

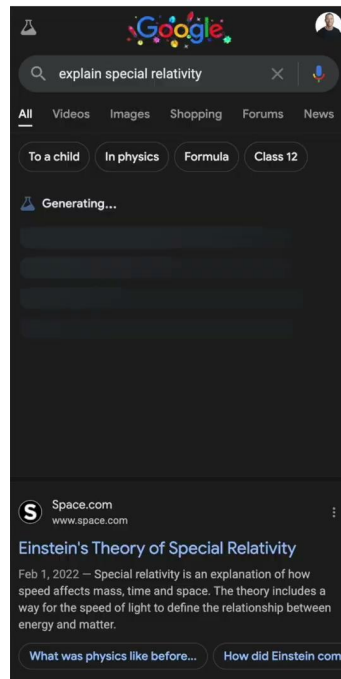
❌ 아무래도 LLM 모델 성능은 클수록 좋지!

❌ 하지만... 이래선 실시간 서비스에서 너무 느려서 쓸 수 없잖아?

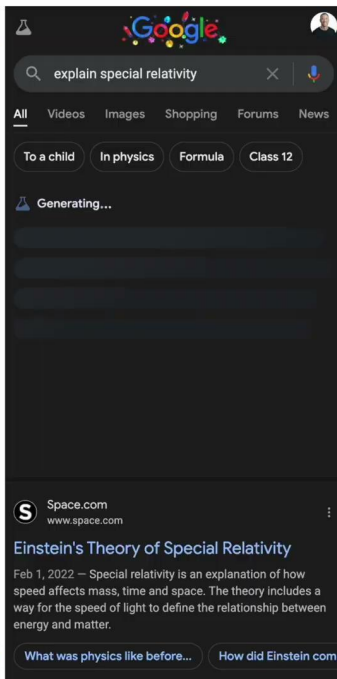
✅ 큰 LLM 모델의 성능은 유지하면서, 좀 더 빠르게 추론이 가능한 방법 없을까?
→ **Speculative Decoding!**

- 빠른 추론 속도
- 추가적인 모델 학습 필요 X
- 빠르지만, 모델 추론 퀄리티는 유지

Before
(no Speculative Decoding)



After
(with Speculative Decoding)

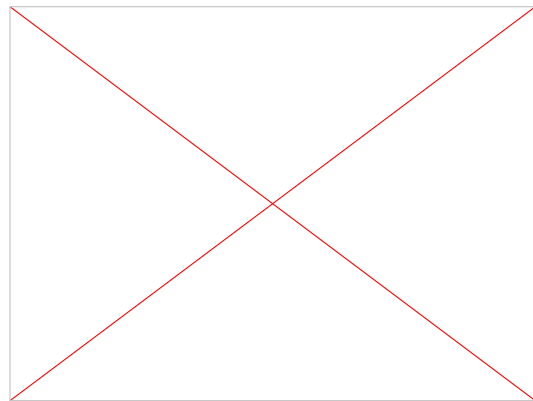


<https://research.google/blog/looking-back-at-speculative-decoding/>

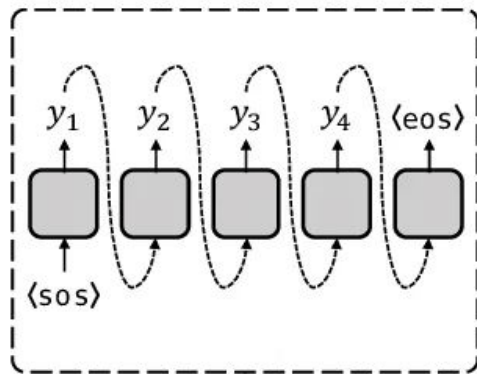
문제1: LLM의 텍스트 생성 방식

1. AutoRegressive Decoding

- LLM은 AutoRegressive Transformer 기반 모델
- AutoRegressive?
 - 이전까지 생성한 토큰(단어)을 기반으로 다음 토큰을 하나씩 예측(생성)
 - 즉, 한 글자, 한 단어씩 **순차적으로 예측**
- 다음 단어를 예측하려면, 이전 단어들이 필요
 - 추론(생성)시에 **병렬처리가 힘들다.**
 - ⇒ **따라서 추론 속도가 느리다**



<https://huggingface.co/blog/assisted-generation#language-decoder-forward-pass-revisited>



<https://medium.com/@peechapp/text-to-speech-models-part-2-autoregressive-models-world-636d8aa0932d>

문제1: LLM의 텍스트 생성 방식

2. Token 별 예측 난이도 차이

What is the square root of 7? The square root of 7 is 2.646

- 7
 - 예측 난이도: 하
 - Input으로 들어간 이전 단어들 중 “the square root of 7”의 패턴이 반복되는 부분이기 때문
 - 의문점: 이걸 거대 LLM으로 추론하긴 좀… 작은 경량급 LLM도 잘 예측할텐데…?
- 2.646
 - 예측 난이도: 상
 - 이전 단어들에 반복되는 패턴 X → 직접 정보를 추출해서 계산을 해야하는 부분이기 때문
 - 의문점: 이 정도 난이도라면, 거대 LLM으로 추론을 해도 되겠군.

문제2: LLM 추론 속도 제한 원인

1. 메모리 대역폭 병목 현상 (Memory Bandwidth Bottleneck)

- 메모리 대역폭(memory bandwidth)
 - 초당 전송할 수 있는 데이터의 양 (VRAM → GPU)
- LLM의 가중치(Weights)는 VRAM에 저장(로딩은 한 번만)
 - LLM은 한 번의 추론시에 모델 전체의 가중치를 반복적으로 읽어야 함 (VRAM → GPU; 읽어오기 != 로딩하기)
 - 예시: 모델 크기가 1TB라면, 매번 1TB의 데이터를 읽어야 새로운 토큰을 생성
 - 이 과정에서 GPU의 연산 성능은 충분하지만,
 - 메모리에서 가중치를 읽어오는 속도가 부족하여 연산 자원이 놀게 된다

문제2: LLM 추론 속도 제한 원인

1. 메모리 대역폭 병목 현상 (Memory Bandwidth Bottleneck)

- GPU/TPU의 높은 연산 처리 능력
 - 1바이트(byte)를 읽을 때 수백~수천 개의 연산(operations) 수행
 - 수백조(trillions) 번의 연산을 1초에 수행 가능
 - 현재의 하드웨어는 매우 높은 연산 처리 능력 보유
- Transformer의 연산 처리
 - 추론 시, 1바이트를 읽을 때 겨우 10번 정도의 연산만 수행
 - Transformer는 하드웨어가 처리할 수 있는 수백~수천 개의 연산 중 아주 일부만 사용

⇒ 즉, 일반적인 GPU 연산에서는 1바이트의 데이터를 읽어오면 수백 번의 연산을 수행할 수 있지만, LLM 추론에서는 한 바이트당 연산량이 매우 적어 GPU가 제대로 활용되지 않는 현상이 발생

- 연산 처리 능력에 반해 메모리 대역폭은 그보다 100배~1000배 낮다(=메모리에서 데이터를 읽는 속도가 비교적 느림)
⇒ 결론적으로 LLM의 추론이 느린 것은 연산이 아니라 모델 가중치를 메모리에서 읽어오는 속도 부족 때문!
⇒ 그럼 연산 처리 자원은 충분히 남는 거네! 메모리 문제만 해결하면 추론 속도 향상을 위해 병렬 처리도 가능하겠는걸?

Speculative Decoding은 문제를 어떻게 해결할까?

Speculative Decoding: 작은 모델(빠른 근사 모델)이 먼저 여러 개의 후보 토큰을 생성하고, 큰 모델이 이를 검토하는 방식

1. 큰 모델의 호출 횟수를 줄여줌

- 기존: 매 토큰 생성마다 큰 모델이 호출됨 → 매번 1TB 가중치를 불러와야 함 (메모리 부담)
- Speculative Decoding:
 - 작은 모델이 먼저 여러 개의 후보 토큰을 생성
 - 큰 모델이 한 번의 호출로 여러 개의 후보를 확인 (병렬 처리)
 - 큰 모델 호출 횟수가 줄어듦 (메모리 대역폭 병목 현상 줄어듦)

2. 메모리 읽기와 연산을 더 균형 있게 사용

- 일반적인 LLM 추론에서는 메모리 읽기 > 연산이라서 연산 유닛(GPU/TPU)이 놀고 있음
- Speculative Decoding을 쓰면 작은 모델이 먼저 여러 개의 후보를 던져주기 때문에, 큰 모델이 더 많은 연산을 한 번에 병렬적으로 수행할 수 있음
- 즉, 메모리 읽기 시간 대비 연산이 더 많아지면서 병렬성 활용이 가능해짐

3. 작은 모델은 연산 비용이 낮고 빠름

- 작은 모델은 가중치 크기가 작고 계산이 빠르기 때문에, 메모리에서 불러오는 부담이 훨씬 적음
- 예를 들어, 큰 모델이 1TB 가중치를 읽어야 한다면, 작은 모델은 10GB 수준일 수도
- 작은 모델이 미리 예측한 토큰이 맞으면, 큰 모델은 추가 계산 없이 바로 다음 단계로 넘어갈 수 있음.

용어

작은 모델

근사 모델

효율적인 모델

M_q

$q(x)$

- 근사 모델에서 t 지점 이전의 토큰들이 주어졌을 때, t 지점의 토큰이 나올 확률 분포

γ (감마)

- 근사 모델(M_q)로 생성할 예측의 개수

큰 모델

타겟 모델

정확한 모델

M_p

$p(x)$

- 타겟 모델에서 t 지점 이전의 토큰들이 주어졌을 때, t 지점의 토큰이 나올 확률 분포

Speculative Decoding

[START] japan ' s benchmark ~~bene~~ n
[START] japan ' s benchmark nikkei 22 ~~7~~ 5
[START] japan ' s benchmark nikkei 225 index rose 22 ~~7~~ 6
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~7~~ 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 ~~7~~ in
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in ~~tekye~~ 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]

[Fast Inference from Transformers via Speculative Decoding](#) (Google, 2022.11)

- 각 라인 → 한 번의 알고리즘 반복(iteration)
- 초록색(✓) 토큰 → 작은 모델이 제안했고, 타겟 모델이 수락한 토큰
- 빨간색(✗) 토큰 → 작은 모델이 제안했지만, 타겟 모델이 거부한 토큰
- 파란색(●) 토큰 → 타겟 모델이 거부한 후, 자체적으로 수정한 토큰
 - e.g. 첫 번째 라인에서는 타겟 모델이 단 한 번 실행되었고, 5개의 토큰이 생성

Speculative Decoding

[START] japan ' s benchmark bond n

$\hat{y}_1 = [\text{START}]$, $\hat{y}_2 = \text{"japan"}$, $\hat{y}_3 = \text{"'"}'$, $\hat{y}_4 = \text{"s"}$, $\hat{y}_5 = \text{"benchmark"}$, $\hat{y}_6 = \text{"bond"}$

1. “[START]” 를 Input으로 받아
2. **근사 모델**이 AutoRegressive Decoding 기법으로 “[START] japan’s benchmark bond”까지 **5개의 토큰을 생성**
 - 근사 모델은 그 크기가 작기 때문에, AR Decoding이더라도, 추론을 빠르게 실행할 수 있음
3. **타겟 모델**에 이를 기반으로 생성한 5개의 토큰을 기반으로 6개의 문장을 병렬로 넣어준다
 - [START]
 - [START] japan
 - [START] japan’
 - [START] japan’s
 - [START] japan’s benchmark
 - [START] japan’s benchmark bond
4. **타겟 모델**은 근사모델에서 각 토큰들이 올바르게 예측된 것인지 병렬적으로 검증하고, 해당 토큰을 Accept/Reject 한다
 - Reject 된 부분 부터 그 이후의 모든 토큰들은 틀린 것으로 간주
 - 해당 부분부터 다시 생성
 - 이미 어떤 토큰을 검사할지 다 알고 있기 때문에, 타겟모델에서 병렬적으로 검사가 가능

$$P_{\text{target}}(\hat{y}_2 \mid \hat{y}_1), \quad P_{\text{target}}(\hat{y}_3 \mid \hat{y}_{1:2}), \quad \dots, \quad P_{\text{target}}(\hat{y}_6 \mid \hat{y}_{1:5})$$

Speculative Decoding

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.

▷ Sample γ guesses x_1, \dots, x_γ from M_q autoregressively.

for $i = 1$ **to** γ **do**

$q_i(x) \leftarrow M_q(prefix + [x_1, \dots, x_{i-1}])$

$x_i \sim q_i(x)$

end for

▷ Run M_p in parallel.

$p_1(x), \dots, p_{\gamma+1}(x) \leftarrow$

$M_p(prefix), \dots, M_p(prefix + [x_1, \dots, x_\gamma])$

▷ Determine the number of accepted guesses n .

$r_1 \sim U(0, 1), \dots, r_\gamma \sim U(0, 1)$

$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$

▷ Adjust the distribution from M_p if needed.

$p'(x) \leftarrow p_{n+1}(x)$

if $n < \gamma$ **then**






$p'(x) \leftarrow \text{norm}(\max(0, p_{n+1}(x) - q_{n+1}(x)))$

end if

▷ Return one token from M_p , and n tokens from M_q .

$t \sim p'(x)$

return $prefix + [x_1, \dots, x_n, t]$

- $\gamma = 5$ (즉, 빠른 모델이 5개의 후보 토큰을 생성)
 - 각 토큰 x_i 에 대해 확률 비교:
 - $r_1 = 0.3, \frac{p_1(x)}{q_1(x)} = 0.4 \rightarrow r_1 \leq 0.4$ 이므로  **Accepted**
 - $r_2 = 0.5, \frac{p_2(x)}{q_2(x)} = 0.6 \rightarrow$  **Accepted**
 - $r_3 = 0.7, \frac{p_3(x)}{q_3(x)} = 0.5 \rightarrow$  **Rejected**
 - $r_4 = 0.2, \frac{p_4(x)}{q_4(x)} = 0.3 \rightarrow$  **Accepted**
 - $r_5 = 0.9, \frac{p_5(x)}{q_5(x)} = 0.8 \rightarrow$  **Rejected**
 - 최종적으로 승인된 토큰 개수 n 계산
 - 거절된 토큰의 인덱스: $\{3, 5\}$
 - 이 정보를 기반으로 최소값 계산 $\rightarrow n = 2$
 - 즉, 첫 2개 토큰만 유지!
- \Rightarrow Rejected된 토큰 이후는 모두 Rejected 이기 때문!

Speculative Decoding

Algorithm 1 SpeculativeDecodingStep [Q]

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

• $P(x)$: 정확한 모델 (M_p)이 예측한 확률

• $Q(x)$: 빠른 모델 (M_q)이 예측한 확률

• $\frac{P(x)}{Q(x)}$:
 ↑ 빠른 모델의 예측이 정확한 모델의 기증에 부합함
 ↓ 빠른 모델의 예측이 정확한 모델의 기증에 부합하지 않음

• 랜덤값 r_i : $0 \sim 1$ 사이의 값을

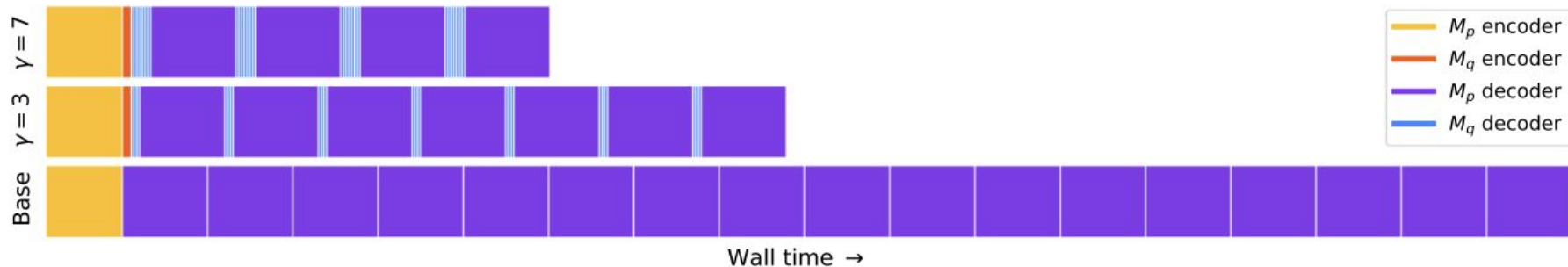
각 후보 토큰 x_i 에 대해 생성
 → r_i 가 $\frac{P(x)}{Q(x)}$ 확률 비율을 넘지 않으면, Accept
 넘으면, Reject

$\therefore r_i \leq \frac{P(x)}{Q(x)}$
 Accept!

$r_i > \frac{P(x)}{Q(x)}$
 Reject!

→ 빠른 모델 (M_q)의 예측이 신뢰할 만 할까?에 대한 검증.

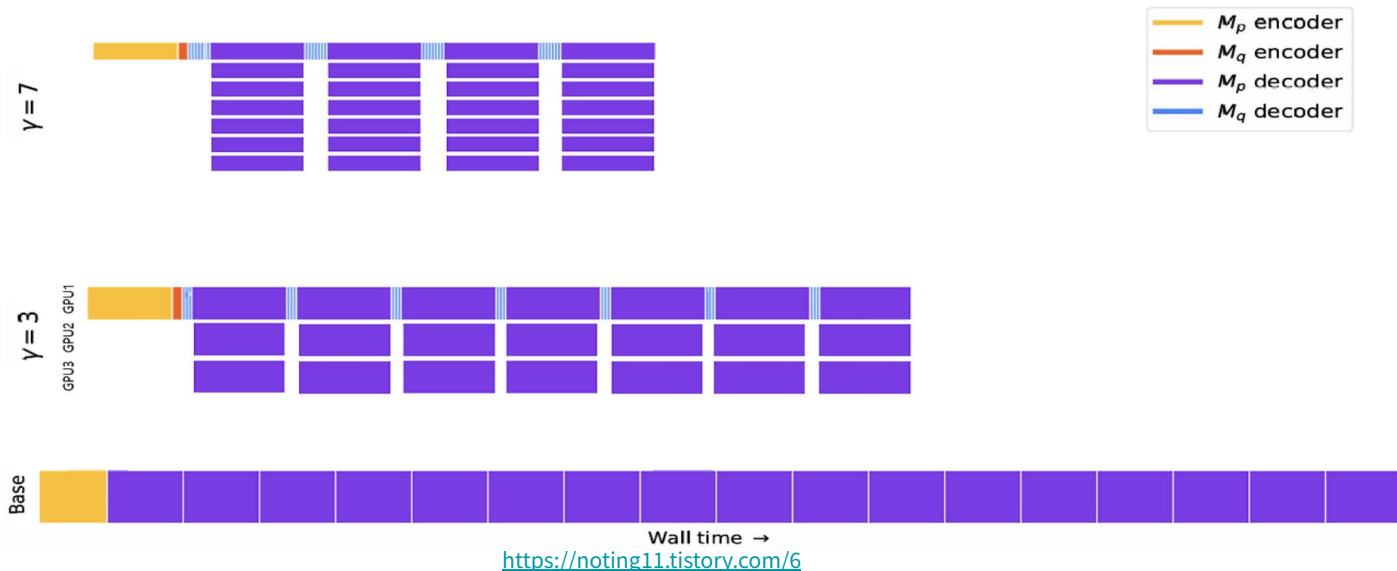
결과: 추론 속도 향상



<https://arxiv.org/pdf/2211.17192>

- **Wall time:** 코드 실행 또는 프로세스가 진행되는 데 걸리는 실제 경과 시간
 - γ : 근사 모델(M_q)로 생성할 예측의 개수 (Speculative Decoding 시행)
 - **파란 부분:** 근사 모델이 정해진 개수의 예측을 생성하는 부분
 - **보라색 부분:** 근사 모델이 생성한 예측에 대해 타겟 모델이 병렬적으로 검증하는 부분
- ⇒ γ 가 7인 Speculative Decoding이 일반적인 순차적 추론 Decoding보다 프로세스 진행 속도가 훨씬 빠르다.

결과: 추론 속도 향상



- 근사 모델(M_q)는 순차적으로 정해진 값(γ) 만큼 추론 진행
- 근사 모델의 추론 결과(γ 개)에 대한 타겟 모델(M_p)의 확률 분포 확인(검증) 진행
 - 이미 타겟 모델의 추론에 필요한 정보는 모두 나와있는 상태.
 - 따라서 앞 추론의 결과를 Input으로 넣기 위해 기다리지 않아도 되므로,
 - 병렬적으로 타겟 모델의 검증을 실행

결과: 추론 속도 향상

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

- TASK
 - ENDe: 영어 → 독일어 번역 (Machine Translation)
 - CNNDM: CNN/Daily Mail 요약 (Summarization)
- M_q (빠른 모델, Proposal Model)
 - T5-SMALL, T5-BASE, T5-LARGE 등 다양한 크기의 모델을 사용
- TEMP (Temperature)
 - 0: Deterministic 방식 (확률적 변동 없음)
 - 1: Stochastic 방식 (확률적으로 변동)
- γ (Guess 수)
 - 빠른 모델(M_q)이 한 번에 예측하는 토큰 수
- α (Acceptance Rate, 승인 비율)
 - 빠른 모델의 제안이 최종적으로 채택될 확률
- SPEED
 - 기존 방식 대비 속도 향상 비율

결과: 추론 속도 향상

- 결과
 - T5-SMALL 모델을 사용하면 가장 큰 속도 향상
 - EnDe 작업에서 T5-SMALL(★) + TEMP 0 조합이 3.4X 속도 증가
 - CNNDM 작업에서도 T5-SMALL(★) + TEMP 0이 3.1X 속도 증가
 - Speculative Decoding을 사용하면 T5-XXL 11B 모델의 속도를 최대 3.4X까지 향상 가능
 - Temperature가 0일 때(TEMP=0), 속도 향상이 더 큼
 - 확률적 샘플링(TEMP=1)이 적용되면 검증 과정이 추가되므로 속도가 다소 감소
 - 큰 모델(M_q = T5-LARGE)을 사용하면 속도 향상 효과가 줄어들음
 - 작은 모델(M_q = T5-SMALL)을 빠른 모델로 사용하면 속도 향상 효과가 큼
 - T5-SMALL > T5-BASE > T5-LARGE 순으로 속도가 빠름
 - 즉, 작은 모델일수록 빠르게 예측하므로 speculative decoding에 적합
 - Acceptance Rate (α)는 모델 크기에 따라 증가
 - T5-SMALL → 낮은 α (ex: 0.75)
 - T5-LARGE → 높은 α (ex: 0.82)

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

참고자료

- [Fast Inference from Transformers via Speculative Decoding](#) (Google, 2022.11)
- [Accelerating Large Language Model Decoding with Speculative Sampling](#) (Deepmind 2023.02)
- <https://research.google/blog/looking-back-at-speculative-decoding/>
- <https://huggingface.co/blog/assisted-generation>
- <https://noting11.tistory.com/6>
- <https://www.youtube.com/watch?v=Sv9bhDYOGcg>
- <https://medium.com/ai-science/speculative-decoding-make-llm-inference-faster-c004501af120>