Trie 기반 KV-cache 공유 검증을 활용한 Lookahead Decoding 가속

이원태, *윤수연
국민대학교, *국민대학교
lwt1025@kookmin.ac.kr, *1104py@kookmin.ac.kr

Accelerating Lookahead Decoding via Trie-based KV-cache Sharing Verification

Won Tae Lee, Soo Yeon Yoon*
Kookmin Univ., *Kookmin Univ.

요약

본 연구는 단일 언어 모델 기반 Lookahead Decoding의 검증(verification) 효율성을 극대화하기 위해, 후보 n-gram을 접두어 트라이(prefix trie)로 재구성하고 KV-cache를 공유하는 새로운 검증 기법을 제안한다. 기존 방식은 후보들을 독립적으로 취급하여 공통 접두어에 대한 중복 연산이 발생하는 한계가 있다. 이에 반해, 제안 기법은 공통 접두어를 트라이 상의 단일 경로로 병합하여 중복 검증을 원천적으로 제거함으로써, 검증 복잡도를 후보 개수가 아닌 트라이의 고유 노드 수에 비례하도록 최적화한다. 실험 결과, 제안 방법은 원본 모델의 생성 품질(Greedy exact match)을 완벽히 유지하면서도 검증 연산량을 유의미하게 감소시켰으며, 이를 통해 전체 디코딩 처리량(Throughput)을 향상시킴을 입증한다.

I. 서론

대규모 언어 모델의 자기회귀(autoregressive) 디코딩은 토큰 단위의 순차적 생성 특성으로 인해 추론 지연(latency)의 주요 병목으로 작용한다. 이를 완화하기 위해, 여러 토큰을 미리 생성한 뒤 검증하는 방식의 Speculative Decoding이 제안되었으며 [1], 이후 단일 언어 모델만을 사용하는 Lookahead Decoding이 제안되었다 [2].

Lookahead Decoding은 여러 후보 n-gram을 병렬적으로 생성하고 검증함으로써 디코딩 step 수를 줄이는 효과적인 가속 기법이다. 그러나 각 후보를 독립적으로 검증하는 구조를 유지하기 때문에, 후보 간에 공통 접두어(prefix)가 존재하더라도 중복된 검증 연산이 반복적으로 발생한다.

본 연구는 이러한 비효율성이 후보 생성 방식이 아닌 검증 구조 자체에서 기인함에 주목한다. 이에 따라 후보 n-gram을 접두어 기반 트라이(prefix trie)로 재구성하고, KV-cache를 공유하는 효율적인 검증 기법을 제안한다.

II. 관련 연구

디코딩 가속을 위한 기존 연구는 자기회귀 디코딩의 순차적 병목을 완화하기 위해 여러 토큰을 병렬적으로 생성하고 검증하는 방향으로 발전해왔다. Lookahead Decoding은 단일 언어 모델만을 사용하여 여러 후보 n-gram을 병렬적으로 생성하고 검증하는 대표적인 접근이다 [2].

Lookahead Decoding은 보조 모델 없이 병렬성을 확보할 수 있다는 장점을 가지지만, 생성된 후보들을 개별적으로 검증하는 구조를 유지한다. Prompt Lookup Decoding(PLD) [3]은 과거 컨텍스트의 접두어를 활용하여 KV-cache를 재사용함으로써 단일 모델 기반 디코딩 가속을 달성한 또 다른 접근이다. 그러나 이들 방법은 공통적으로, 동일 디코딩 step에서 생성된 후보들 간의 검증 중복을 구조적으로 제거하지는 않는다.

본 연구는 이러한 한계에 주목하여, 후보 시퀀스를 접두어 기반 트라이 구조로 재구성하고 공통 접두어에 대한 KV-cache를 공유하는 Trie 기반 검증 기법을 제안한다.

III. 실험 설계

3.1 Trie 기반 KV-cache 공유 검증

Lookahead Decoding의 검증 단계에서 발생하는 중복 연산을 제거하기 위해, 후보 n-gram을 접두어 기반 트라이(prefix trie)로 재구성하고 KV-cache를 공유하는 검증 기법을 제안한다. Algorithm 1은 제안하는 트라이 기반 KV-cache 공유 검증 절차를 나타낸다. 여기서 D는 단일 후보 경로의 최대 검증 깊이, B는 한 스텝의 최대 검증 노드 수를 의미한다.

Algorithm 1 Trie-based KV-shared Verification

Require: $KV(x_{1:t})$, Candidates C, Limits D, B
Ensure: Accepted sequence y
1: 후보 집합 C로부터 트라이 T를 구성하고, 예산(D, B) 내에서 부분트리 T'를 선택
2: **for** $v \in V(T')$ in topological order **do**
3: **if** $KV(v)$ exists **then** Reuse **else** Compute & Cache $KV(v)$
4: **end for**
5: Greedy 예측과 일치하는 T' 상의 최장 경로를 수용하여 y 생성
6: 컨텍스트 및 KV-cache 갱신

3.2 복잡도 분석

한 디코딩 step에서 N는 후보 n-gram 수, D는 후보당 최대 검증 길이를 의미한다. 후보 집합 $C = \{c_1, \ldots, c_N\}$를 트라이 T로 병합하고, 실제로 탐색되

는 부분트리를 $T' \subseteq T$라 할 때, $|V(T')|$는 부분트리의 고유 노드 수를 의미한다. incremental decoding 특성상, 검증 단계의 비용은 incremental forward 횟수, 즉 검증된 고유 노드 수에 비례한다고 가정한다.

기존 Lookahead Decoding은 각 후보별로 최대 D개 토큰을 독립적으로 검증하므로, 검증 복잡도는 $\mathcal{O}(N \cdot D)$이다. 반면 제안 기법은 동일한 접두어를 한 번만 검증하므로, 검증 비용은 후보 수가 아닌 부분트리의 고유 노드 수에 비례하며 $\mathcal{O}(|V(T')|)$로 표현된다. 접두어 공유가 많은 경우 일반적으로 $|V(T')| \ll N \cdot D$가 성립한다. 여기서 각 노드 v에 대한 KV 계산은 부모 접두어의 KV-cache를 기반으로 한 single-token incremental forward에 해당한다.

3.3 비교군 및 하이퍼파라미터

제안 기법의 성능은 다음의 세 가지 디코딩 방식과 비교하여 평가한다.

(i) **Autoregressive (Greedy):** 토큰 단위의 순차적 생성 방식.

(ii) **Lookahead Decoding (Baseline):** 원본 Lookahead Decoding 방식으로, 후보 n-gram을 독립적으로 검증하며 공식 구현체(LADE)의 설정을 따른다.

(iii) **Trie-based Lookahead (Ours):** 본 연구에서 제안하는 트라이 기반 KV-cache 공유 검증 방식.

모든 디코딩 방식에서 Temperature는 0.0으로 고정하여 결정론적(deterministic) 결과를 보장한다. Lookahead 계열의 기본 설정은 윈도우 크기 $W = 10$, 후보 가지 수 $N = 5$, 최대 n-gram 길이 $L = 5$로 설정하였다. 제안 기법은 추가적으로 검증 예산을 제어하기 위해 최대 탐색 노드 수 B를 설정한다.

IV. 실험 결과

4.1 검증 비용(Verification Cost) 분석

Table 1은 동일한 후보 생성 전략 하에서 Baseline Lookahead와 제안하는 Trie-based Lookahead의 검증 비용을 비교한 결과를 보여준다. 검증 비용은 디코딩 과정에서 실제로 연산된 총 토큰 수(Computed Tokens)와, 이를 생성 토큰 수로 정규화한 Cost/Token으로 측정한다.

Table 1. 디코딩 방식별 생성 토큰 수 및 검증 비용 비교

Method	Gen. Tokens	Comp. Tokens
Autoregressive	107	209
Lookahead (Baseline)	256	1,681
Lookahead (Ours)	**257**	**772**

Autoregressive 방식은 EOS 토큰 발생 시 즉시 종료하였으며, Lookahead 계열은 검증 비용의 추세를 관찰하기 위해 최대 생성 길이 $T = 256$까지 생성을 지속하였다.

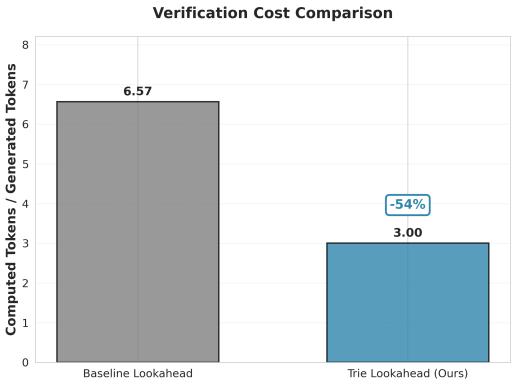

Fig. 1. 디코딩 방식별 Cost/Token 비교

Baseline Lookahead는 256개의 토큰을 생성하는 동안 총 1,681개의 토큰을 연산한 반면, 제안 기법은 772개의 토큰만을 연산하여 Fig 1과 같이 **검증 비용을 약 54% 감소**시켰다.

4.2 처리량에 대한 논의

Python 기반 실험 환경의 오버헤드로 인해 시간적 처리량의 이득은 제한적이었으나, Computed Tokens가 절반 이하로 감소했다는 점은 최적화된 서빙 환경에서 지연 시간 감소로 이어질 가능성을 시사한다.

V. 결론

본 연구는 Lookahead Decoding의 검증 단계를 트라이 기반 KV-cache 공유 구조로 재구성하여, 공통 접두어로 인한 중복 연산을 제거하는 기법을 제안하였다. 제안 방법은 출력 품질을 유지하면서도 검증 비용을 유의미하게 감소시켰으며, 이를 통해 단일 모델 기반 디코딩 가속이 후보 생성 방식뿐 아니라 검증 구조의 설계에 의해서도 효과적으로 개선될 수 있음을 보였다. 또한 이는 기존 Lookahead 계열 기법 뿐만 아니라 접두어 공유가 발생하는 다양한 추론 가속 시나리오로 확장 가능하다는 점에서, 효율적인 추론 구조 설계를 위한 다양한 방향의 초석이 될 수 있다.

References

[1] Y. Leviathan, M. Kalman, and Y. Matias, "Fast Inference from Transformers via Speculative Decoding," *arXiv preprint arXiv:2211.17192*, 2023.

[2] Y. Fu, P. Bailis, I. Stoica, and H. Zhang, "Break the Sequential Dependency of LLM Inference Using Lookahead Decoding," *arXiv preprint arXiv:2402.02057*, 2024.

[3] Y. Chen et al., "Prompt lookup decoding for large language models," *arXiv preprint arXiv:2307.xxxx*, 2023.