KV 캐시 스토리지 오프로딩에서 유사 프롬프트의 캐싱 방법에 관한 연구

이청솔
삼성전자

cs89.lee@samsung.com

A Study on the Caching Strategies for Semantically Similar Prompts in KV Cache Storage Offloading

Lee Choungsol

Samsung Electronics

요 약

스토리지 오프로딩 기반 LLM 캐시는 일반적으로 입력 토큰의 정합 일치(exact match)에 대해서만 KV 캐시 재사용을 허용해 실제 환경의 다양한 표현 변형에는 한계가 있다. 본 연구는 이러한 한계를 극복하기 위해 의미적으로 유사한 프롬프트까지 재사용 범위를 확장하는 유사도 기반 KV 캐시 재사용 기법을 제안한다. 공개 데이터셋을 활용한 실험에서 제안 기법은 응답 품질을 유지하면서도 TTFT 를 단축시켰으며, 특히 입력이 길수록 효과가 커져 TTFT 가 최대 80.4% 감소하였다. 본 기법은 추가 학습 없이 기존 서빙 스택과 호환되어 실서비스 환경에서 실용적 이득을 제공한다.

Ⅰ. 서 론

대규모 언어 모델(LLM)에서 KV 캐시를 활용한 성능 최적화가 중요한 연구 주제로 떠오르고 있다. LLM 추론의 총 지연시간(latency)은 프리필(입력 토큰 처리) 단계와 디코드(출력 토큰 생성) 단계로 구성되며, 특히 긴 컨텍스트를 처리할 경우 TTFT(Time To First Token)가 체감 성능을 좌우한다.

일반적으로 프리필 단계에서 KV 캐시를 GPU 메모리에서 찾아 사용하는데, 근래 연구에서는 GPU 메모리의 크기 한계로 인해 CPU 메모리나 스토리지로 오프로딩 하는 연구가 이루어지고 있다. 현재 대표적인 스토리지 오프로딩 사례로는 Deepseek 의 context caching[1]이 있으며, 캐싱 전략은 exact match 를 기반으로 한 KV 캐시 재사용이다. 그러나 원격 저장소 환경에서 다수의 사용자가 KV 캐시를 공유하는 상황이나, 단일 환경이라도 유사 프롬프트를 빈번하게 사용하는 등 실제 응용 상황을 고려할 때 exact match 방식은 한계가 있다. 본 논문에서는 문장 간 유사도가 높은 경우에도 캐시를 매칭하여 KV 캐시를 재사용하는 방법을 제안하며, 이를 통해 프리필 단계의 응답 지연 감소와 시스템 처리 효율을 향상시키는 것을 목표로 한다.

이를 해결하기 위해 본 연구는 (1) 유사문장이 입력되면 스토리지에 저장된 프롬프트 임베딩을 벡터 검색으로 찾아 해당 프롬프트에 매칭된 재사용할 KV 캐시를 선택하고, (2) 벡터 검색으로 찾은 원본 프롬프트의 토큰열을 그대로 사용하도록 입력을 대체하여, 최종적으로 모델에 주입되는 토큰열이 캐시 생성 시점과 정확히 동일해지도록 함으로써 exact match 기반 KV 캐시 재사용을 달성한다. 논문의 핵심 질문은 다음과 같다.

Q1. 스토리지에 저장된 KV 캐시를 재사용하면 TTFT 가 감소하는가?

Q2. TTFT 감소효과는 모델 및 입력길이에 따라 달라지는가?

Q3. 유사 프롬프트를 원본문장으로 정규화하여 KV 캐시를 재사용할 경우, 직접 인퍼런스를 수행한 결과와 의미적으로 동등한가?

Ⅱ. 본론

2.1 제안 기법 개요 본 연구에서는 문장 간 의미 유사도에 기반한 KV 캐시 매칭 알고리즘을 제안한다. 실제 서비스 환경에서 사용자가 생성하는 다양한 프롬프트를 최대한 대체할 수 있도록 여러 데이터셋을 활용하였다. 데이터셋의 문장을 패러프레이즈하여 유사문장을 생성한 뒤 실험을 진행하였으며, 검색 결과의 유사도가 일정 기준 이상인 경우 해당 원본문장의 KV 캐시를 스토리지에서 불러와 프리필 단계에서 재사용하였다.

2.2 실험 방법 먼저, vLLM 서버에 LMCache[2]를 연동하여 각 데이터셋의 문장을 입력하고 프리필을 수행하여 KV 캐시를 생성한 뒤 디스크에 저장하였다. 이어서 동일 문장의 임베딩을 계산하여 FAISS[3] 인덱스에 등록하였다. 이후 원본문장을 대규모 언어 모델로 패러프레이징 하여 단어 및 구문을 변형함으로써 exact match 가 성립하지 않도록 하였다. 패러프레이징 시 문장 생성 temperature 는 0.1 부터 1.0 까지 검증한

뒤, 의미 보존과 다양성의 균형을 고려하여 0.3 을 최종 설정하였다. 이렇게 생성된 유사문장은 임베딩 후 벡터 검색을 수행하여 Top-1 후보를 도출하고, 검색된 원본문장과 질의 간의 코사인 유사도가 사전에 설정한 임계치 이상일 경우 해당 원본문장의 KV 캐시를 디스크에서 불러와 프리필 단계에서 재사용하였다. 마지막으로, 정확도 평가는 각 문단에 대한 질문을 모델에 제시하여 출력 응답과 데이터셋의 정답을 비교하고, F1 및 EM 지표를 계산함으로써 수행하였다.

그림 1. 실험 개요

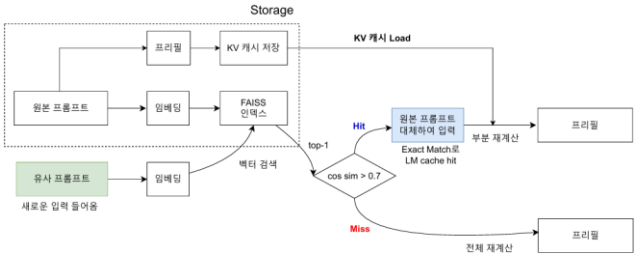

그림 1 은 제안 기법의 전체 실험 파이프라인 개요이다. 유사 프롬프트 입력 시 벡터 검색과 코사인 유사도 임계치(0.7)를 통해 원본 프롬프트로 정규화하고, 해당 KV 캐시를 로드하여 재사용한다.

실험에는 NVIDIA H100 80GB GPU, Intel(R) Xeon(R) Platinum 8480+ CPU, 삼성전자 서버용 DDR5 RAM, 삼성전자 Gen5 SSD 를 사용하였다.

표 1. Llama 3.1-8B 데이터셋별 F1/EM

데이터셋	특징	토큰	F1	EM
SQuAD	Wiki	150	0.81	0.63
QuAC	Wiki	400	0.72	0.6
NewsQA	CNN	800	0.61	0.37
TriviaQA	Wiki	10k	0.54	0.2

표 1 은 유사문장 벡터서치 및 KV 캐시를 재사용한 출력과 데이터셋의 정답을 비교한 것이다. 높은 F1 점수로 미루어 보아, 유사문장 입력시 원본문장과 매칭이 잘 이루어지고 있음을 알 수 있다. 토큰수가 적은 데이터셋일 수록 정확한 정답을 얻을 수 있었다.

그림 2. 데이터셋 별 TTFT 감소 효과 비교

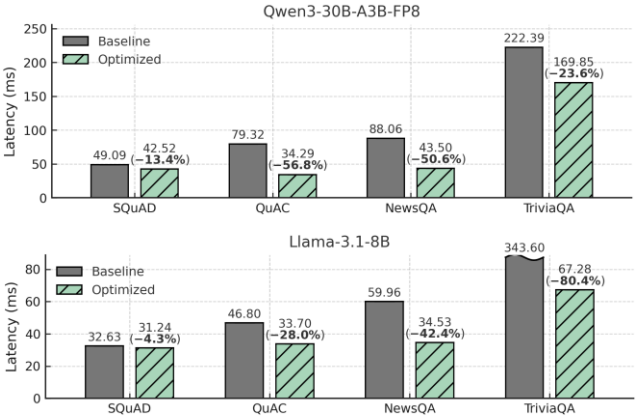

그림 2 에서 보이듯 TTFT 감소는 데이터셋의 입력 토큰 수가 증가할수록 더욱 뚜렷하게 나타났다. 이는 입력이 길어질수록 프리필 비용이 커지기 때문에, 캐시 재사용에 따른 상대적 이득이 확대되었기 때문이다. 다만 Qwen3-30B 모델은 가장 긴 입력에서 TTFT 감소율이 낮게 관찰되었다. 이는 해당 모델이 유사문장 생성 시 표현 다양성이 높아 miss 발생 빈도가 늘어났고, miss 가 발생할 경우 전체 프리필을 다시 수행해야 하므로 평균 지연시간 개선폭이 줄어든 것으로 추정된다.

Ⅲ. 결론

본 논문에서는 기존 exact match 기반 KV 캐시 재사용의 한계를 극복하기 위해 문장 간 의미 유사도 기반의 KV 캐시 매칭 기법을 제안하였다. KV 캐시를 디스크에 저장하고 불러오는 과정에서 TTFT 이득을 얻었고, 벡터검색과 코사인 유사도 검증으로 잘못된 캐시 로드를 방지해 신뢰성을 높였다. 서론에서 이야기한 논문의 핵심 질문에 대한 대답은 다음과 같다.

Q1. 스토리지에 저장된 KV 캐시를 재사용하면 TTFT 가 감소하는가? 모든 데이터셋에서 TTFT 가 유의미하게 감소하였다.

Q2. TTFT 감소 효과는 모델 및 입력 길이에 따라 달라지는가? TTFT 절감은 입력 길이에 따라 뚜렷하게 달랐다. 짧은 입력은 4% 수준에 그쳤으나, 10k 토큰 입력에서는 최대 80.4%까지 감소하였다. 다만 Qwen3-30B 모델은 miss 발생률이 높아 대규모 입력에서 개선폭이 제한적이었다.

Q3, 유사 프롬프트를 원본문장으로 정규화하여 KV 캐시를 재사용할 경우, 직접 인퍼런스를 수행한 결과와 의미적으로 동등한가? 그렇다. 유사문장을 입력해 의미적으로 같은 원본문장을 찾아 대체해 KV 캐시를 재사용 한 경우 원래 데이터셋의 정답과 F1/EM 상 높은 일치도를 보였다.

ACKNOWLEDGMENT

This work was supported by the Samsung Electronics Samsung Memory Research Center (SMRC). The authors would like to express their gratitude to the SMRC research staff for their valuable technical assistance and insightful discussions throughout the course of this study.

참 고 문 헌

[1] DeepSeek, “Context Caching.” Available: https://api-docs.deepseek.com/guides/kv_cache

[2] OpenLM Research, “LMCache: Efficient KV Cache Storage Offloading for LLM Inference,” 2024. Available: https://lmcache.ai/

[3] J. Johnson, M. Douze, and H. Jegou, “FAISS: A library for efficient similarity search and clustering of dense vectors,” 2019. Available: https://github.com/facebookresearch/faiss