LLM 추론 성능 최적화를 위한 클러스터링 기반 KV 캐시 재사용 시스템

김수진 [1], 유헌창 [2]
[1] 고려대학교 SW.AI 융합대학원 인공지능융합학과 석사과정
[2] 고려대학교 정보대학 컴퓨터학과 교수

sujinii.kim@gmail.com, yuhc@korea.ac.kr

A Clustering-based KV-Cache Reuse System for Optimized LLM Inference Performance

Sujin Kim[1], Heonchang Yu[2]
[1]Dept. of Applied Artificial Intelligence, College of SW.AI Convergence, Korea University
[2]Department of Computer Science & Engineering, College of Informatics, Korea University

요 약

대규모 언어 모델(LLM)은 다양한 자연어 처리 과제에서 탁월한 성능을 보이지만, 추론 과정에서 발생하는 높은 연산 비용과 긴 지연 시간은 실시간 응용 분야에 심각한 제약으로 작용한다. 본 논문은 이러한 문제를 해결하기 위해, 질의의 의미적 유사성을 기반으로 KV 캐시를 사전에 생성하고 이를 효율적으로 재사용하는 시스템을 제안한다. 제안 방식은 크게 두 단계로 나누어진다. 우선 전처리 단계에서 Sentence-BERT 임베딩과 Spectral Clustering 을 활용하여 질의를 의미적으로 군집화하고, 각 군집에 대한 대표 캐시를 미리 구축한다. 이후 실시간 추론 시에는 새로운 질문을 이미 생성된 군집 중에서 가장 유사한 군집을 찾아 매핑한다. 캐시를 즉시 불러오는 과정을 통해, 불필요한 LLM 연산을 줄이고 응답 속도를 극대화한다. 실 사용 중인 금융 QA 데이터 셋을 활용하여 실험한 결과, 본 시스템은 기존 Cache-Augmented Generation(CAG) 접근법과 비교하여 눈에 띄는 성능 향상을 보였다. 구체적으로, 응답시간은 99.63% 단축해 평균 3.66 초에서 0.0134 초로 줄였다. 이는 처리량(QPS)을 273 배 증가시킨 74.63QPS 에 해당하는 성능이다. 이러한 성과는 본 논문에서 제안하는 시스템이 대규모 데이터 환경에서도 안정적으로 동작하며, 실용적으로 활용할 수 있음을 보여준다.

1. 서론

대규모 언어 모델(LLM)의 등장은 자연어 처리 분야에 혁신적인 변화를 불러온 것으로 평가된다. 그러나 거대 모델의 크기로 인해 막대한 연산 비용과 추론 지연 시간 문제가 발생하였으며, 이로 인해 대규모 언어 모델을 실시간 서비스에 적용하는 데 한계가 드러났다. 이러한 문제를 해결하기 위한 방안 중 하나로 트랜스포머 아키텍처의 자기회귀(auto-regressive) 디코딩 과정에서 KV 캐시(Key-Value Cache)가 도입되었다. KV 캐시는 이전에 계산된 토큰의 연산 결과를 캐시에 저장하여 반복적인 계산을 줄이는 역할을 한다. 그러나 기존의 Cache-Augmented Generation(CAG)[1] 시스템은 KV 캐시를 활용하지만, 모든 질의를 캐시에 넣어 관리하는 구조를 채택하고 있어, 효율성이 측면에서 한계가 있다. 예를 들어 “AI 의 정의는 무엇이야?”와 “인공지능을 설명해 줘”와 같이 의미상으로 동일한 질의에 대해 매번 새로운 KV 캐시를 생성하고 계산을 수행하기 때문에, 시스템 성능 저하를 일으킨다.

본 연구는 기존 KV 캐시 기법의 비효율성을 극복하기 위해, 질의 간 의미적 유사도를 활용한 사전 캐시 생성 및 재사용 최적화 시스템을 새롭게 제안한다. 핵심 아이디어는 의미가 유사한 질의들은 본질적으로 유사한 KV 캐시 상태를 공유할 수 있다는 가정에 기반한다. 이를 위해 대규모 질의 데이터를 임베딩한 뒤 군집화하고, 각 그룹을 대표하는 질의를 KV 캐시에 사전 생성 후 저장한다. 이러한 전처리 과정을 통하여, 추론 시 모든 계산을 반복하지 않고, 가장 적합한 캐시를 즉시 불러와 응답 속도를 크게 향상할 수 있다. 특히 본 연구에서는 질문의 반복성이 높은 금융 QA 데이터 셋을 활용하여 실험을 진행하였으며, 그 결과 제안한 기법이 단순한 학술적 기여를 넘어 실제 산업 환경에서도 높은 실용적 가치를 지니고 있음을 검증하였다.

2. 데이터 전처리

2.1 KV 캐시 최적화 연구의 현황

LLM(대규모 언어 모델)의 핵심인 트랜스포머 아키텍처는 토큰을 생성할 때마다 전체 입력 시퀀스에 대해 셀프어텐션(Self-Attention)을 재계산해야 하는 비효율성을 가지고 있다. 이러한 문제를 해결하기 위해, 이전에 계산된 키(key)와 값(Value) 쌍을 저장하는 KV 캐시 개념이 활용되었다. KV 캐시는 다음 토큰을 생성할 때 캐시에 저장된 키(key)와 값(Value)을 재활용함으로써, 연산량 감소의 핵심적인 역할을 한다.

기존 연구들은 주로 KV 캐시의 물리적 관리 최적화에 집중해 왔다. 대표적으로 양자화(Quantization)[2]를 통해 캐시 크기를 줄이거나, 사용 빈도가 낮은 캐시를 제거하는 교체 정책(Eviction Policy)연구가 이에 속한다. 그러나, 이러한 방법들은 질의의 의미적 특성을 충분히 고려하지 못한다는 한계가 있다.

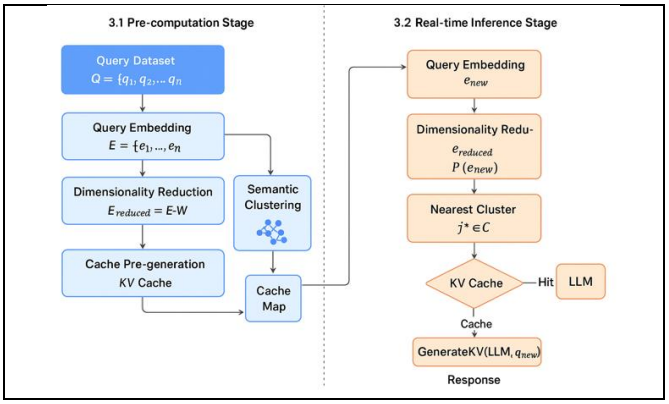
2.2 클러스터링 기반 자연어 처리(NLP) 응용

클러스터링은 데이터 표현을 유사성 기반으로 그룹화하는 대표적인 비지도 학습 기법으로, NLP 분야에서는 문서 요약, 의미적 검색 등 다양한 응용 분야에서 널리 활용되고 있다. 특히 Sentence Transformer 와 같은 고성능 임베딩 모델의 발전으로 문장이나 질의를 고차원 벡터 공간에서 정밀하게 표현할 수 있게 되었으며, 이를 통해 복잡한 의미적 관계를 포착하는 새로운 연구들이 활발히 진행되고 있다.

클러스터링 기법에는 K-Means, GMM(Gaussian Mixture Model), Agglomerative Clustering, Spectral Clustering 등이 있으며, K-Means 나 GMM 은 군집이 선형적으로 구분이 가능한 경우에 효과적이지만, Spectral Clustering 은 그래프 기반 접근을 통해 비정형적 구조를 가진 데이터에서도 뛰어난 성능을 보인다.

3. 의미 기반 캐싱

본 논문에서 제안하는 시스템은 의미 기반 캐싱으로, 전처리 단계(pre-computation stage)와 실시간 추론 단계(real-time inference stage)의 두 단계로 구성된다. 이러한 2 단계 구조를 통해, 대규모 언어 모델의 연산이 최소화되어 비용이 절감되며, 효율성이 극대화 된다.



(그림 1) 의미 기반 캐싱 (좌)전처리 (우) 실시간 추론

3.1 전처리 단계: 의미 기반 캐시 사전 생성

전처리 단계는 대규모 질의 데이터 셋을 분석하여 재사용 가능한 KV 캐시를 미리 생성하는 과정이다. **(1) 질의 임베딩** 단계에서는 대규모 질의 데이터 셋 $Q = \{q1, q2, \ldots, qn\}$에 대해, Sentence-BERT[4]와 같은 고성능 임베딩 모델을 사용하여 각 질의를 고차원 벡터 공간의 임베딩 E={e1 ,e2 ,…,en } 으로 변환한다. 이어서 **(2) 차원축소** 과정에서는 임베딩된 벡터 E 에 주성분 분석(PCA)[5]을 적용하여 차원을 효율적인 수준으로 축소한다. 이 과정은 복잡한 클러스터링 연산의 계산 비용을 대폭 줄이는 핵심적인 전처리 단계이며, 수식은 다음과 같다.

$$E_{reduced} = E \cdot W$$

여기서 $W \in \mathbb{R}^{d \times k}$는 주성분들의 행렬을 의미하고, d는 원본 차원, k는 축소된 차원을 의미한다.

다음으로, **(3) 의미적 군집화** 단계에서는 차원 축소된 임베딩 $E_{reduced}$에 Spectral Clustering[4]을 적용하여 의미적으로 유사한 질의들을 k 개의 군집 $C = \{c1, c2, \ldots, ck\}$으로 나눈다. 마지막으로 **(4) KV 캐시 사전 생성** 단계에서는 각 군집 Cj의 중심점(centroid)에 가장 가까운 실제 질의 q_{rep}를 대표 질의로 선정한다. 이후 해당 대표 질의에 대해 LLM 을 한 번 호출하여 KV 캐시 Kj를 미리 계산하고 이를 저장소에 보관하여 추후 유사 질의에 대한 효율적인 캐시 재사용을 가능하게 한다.

3.2 실시간 추론 단계: 클러스터링 기반 캐시 활용

Algorithm 1: Accelerated Inference with Clustered KV Cache

Input: New query q_{new}
Output: Generated response R

Data Structures:

- M: Pre-generated KV cache map, indexed by cluster centroid
- E: Query embedding model
- P: PCA model for dimension reduction
- K: Cluster centroids $\{K_1, K_2, \ldots, K_k\}$
- LLM: Large Language Model

1. $e_{\text{new}} \leftarrow E(q_{\text{new}})$ // Embed the new query
2. $e_{\text{reduced}} \leftarrow P(e_{\text{new}})$ // Reduce embedding dimension
3. $j^* \leftarrow \arg\min_{j \in \{1,\ldots,k\}} \text{distance}(e_{\text{reduced}}, K_j)$
   // Find nearest centroid
4. **if** $j^* \in M$ // Cache Hit
   a. $KV_{\text{cache}} \leftarrow M[j^*]$
   b. $R \leftarrow \text{Generate}(LLM, KV_{\text{cache}})$
5. **else** // Cache Miss
   a. $KV_{\text{cache}} \leftarrow \text{GenerateKV}(LLM, q_{\text{new}})$
   b. $R \leftarrow \text{Generate}(LLM, KV_{\text{cache}})$
   c. $M[j^*] \leftarrow KV_{\text{cache}}$
6. **return** R

(그림 2) 클러스터링 기반KV 캐시 Algorithm

실시간 추론 단계에서는 새로운 질의가 입력되면, 먼저 **(1) 임베딩 모델**을 통해 고차원 벡터로 변환한다. 이후, 이 벡터는 **(2) PCA 기반 차원 축소** 과정을 거쳐 고차원 데이터를 효율적으로 압축된 저차원 표현으로 변환한다. 이렇게 차원 축소된 임베딩은 사전에 계산된 **(3)클러스터 중심들과 비교**하여 가장 가까운 클러스터를 선택한다.

다음으로, 선택된 클러스터에 대응하는 KV 캐시가 존재하는 경우, 해당 KV 캐시를 즉시 불러와 언어 모델에 전달한다(Cache Hit). 반면, KV 캐시가 존재하지 않는 경우(Cache Miss) 언어 모델을 직접 호출하여 새로운 KV 캐시를 생성하고, 이를 저장소에 보관한 후 응답을 생성한다.

본 논문에서 제안한 시스템은 질의 별로 적합한 캐시를 효율적으로 재사용함으로써, 응답 생성 속도를 크게 향상시키고 계산 비용을 절감할 수 있다.

3.3 기존 Semantic Caching 방식과의 차별성

제안하는 시스템은 의미적으로 유사한 질의를 재활용한다는 점에서 Semantic Caching 연구와 유사하다.

그러나 기존 방식이 실시간 질의-결과 매칭에 의존했던 반면, 본 연구는 **사전 전처리 기반 캐시 생성** 구조를 도입하여 대규모 LLM 환경에서의 연산 효율을 획기적으로 개선한다.

본 시스템의 차별성은 두가지로 요약된다.

첫째, **사전 클러스터링 기반 캐시 구조**를 도입하여, 질의 발생 이전에 대규모 질의 집합을 임베딩·차원축소·군집화함으로써 유사 질의 탐색 과정을 미리 최적화한다. 이를 통해 실시간 질의 처리 시 유사도 계산 부하를 최소화하고, 응답 속도를 크게 향상시킨다.

둘째, **대표 질의 중심의 캐시 압축 매커니즘**을 활용한다. 각 군집의 중심점을 대표 질의로 선정하고 이에 대한 KV캐시만을 사전 계산·저장함으로써, 메모리 사용량을 줄이는 동시에 중복 연산을 방지한다.

이와 같은 사전 지식 기반의 캐시 구조는 기존 Semantic Caching 이 가지던 실시간 매칭의 병목을 제거하고, **효율적이고 확장 가능한 캐시 재사용 프레임워크**를 제시한다는 점에서 차별성이 있다.

4. 실험 결과 및 분석

본 연구의 실험은 NVIDIA A100 GPU 4 대로 구성된 환경에서 DeepSeek 모델을 기반으로 수행되었다. 데이터 셋은 실제 금융 서비스 AI 챗봇 로그를 정제한 Real Data QA 데이터 셋을 사용하였다. 이를 통해 제안 기법의 실용성 및 일반화 가능성을 검증하였다.

제안하는 클러스터링 기반 캐싱 기법의 성능을 다각도로 분석하기 위해, 다음 세 가지 평가 항목을 설정하였다.

(1) 클러스터링 방법론 비교
(2) 하이퍼파라미터 최적화
(3) 데이터 셋 규모에 따른 성능 확장성

4.1 클러스터링 방법론 비교

제안하는 시스템의 핵심은 의미 기반 클러스터링이다. 이 실험에서는 Spectral Clustering 의 우수성을 입증하기 위해 K-Means, GMM, Agglomerative Clustering 과 성능을 비교하였다. 또한, 해당 실험은 데이터 크기 2,000 개, 클러스터 수 8 개 및 PCA 차원 32 로 고정하여 진행하였다.

<표 1> 클러스터링 방법론 비교

알고리즘	평균 응답 시간 (s)	캐시 적중률 (%)	속도 개선율 (%)
Spectral	0.0134	99.60	99.63
Agglomerative	0.42	87.20	88.65
GMM	0.81	75.10	79.43
K-Means	1.15	68.30	68.92

<표 1>은 각 알고리즘의 성능을 비교한 결과를 보여준다. Spectral Clustering 캐시 적중률에서 99.6%를 기록하여 다른 방법들보다 현저히 우수한 성능을 보였으며, 평균 응답 시간 또한, 0.0134 초로 가장 짧게 측정되었다. 반면, K-Means 와 GMM 은 캐시 적중률 및 응답 시간 모두에서 상대적으로 낮은 성능을 보였다. 이러한 차이는 알고리즘의 구조적 특성에 기인한 것으로 해석된다. 특히 K-Means 와 GMM 이 클러스터를 구형(Spherical) 혹은 볼록(convex) 형태로 상정하는 반면, Spectral Clustering 은 데이터 간의 복잡한 연결 관계를 그래프 기반으로 모델링함으로써 자연어 질의가 지니는 비선형적 의미 구조를 효과적으로 반영할 수 있다.

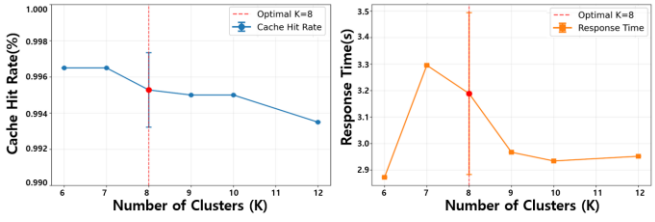
Spectral Clustering 알고리즘의 핵심 과정은 유사성 행렬 A 와 차수 행렬 D 를 기반으로 정규화 라플라시안 행렬을 계산한 뒤, 해당 행렬의 주요 고유 벡터를 추출하여 K-Means 로 분류를 수행하는 것이다. 정규화 라플라시안 행렬은 다음과 같이 정의된다.

$$L_{sym} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$$

이러한 과정을 통해 데이터의 내재적 구조를 보다 충실히 반영할 수 있고, 실험 결과에서 확인된 성능 차이는 이론적 장점이 실제 효과로 직결됨을 보여준다. 따라서, <표 1>에 제시된 수치는 Spectral Clustering 의 구조적 특성과 이로 인한 실질적 성능 향상을 명확히 입증한다고 할 수 있다.

4.2 하이퍼파라미터 최적화

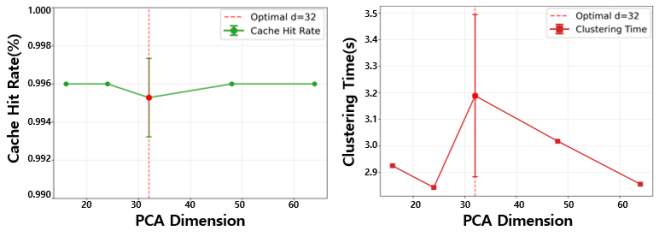
본 시스템의 캐싱 성능을 극대화하기 위해 클러스터 수(k)와 PCA 차원 수(d)를 변수로 설정하여 최적의 하이퍼파라미터 조합을 탐색하는 실험을 수행하였다.



(그림 3) 클러스터 수(k) 비교

(그림 3)은 클러스터 수(k)에 따른 캐싱 성능 변화를 보여준다. 클러스터 수(k)값이 지나치게 작을 경우

의미상 다른 질의들이 동일한 그룹에 포함되어 캐시 적중률이 저하되는 경향을 보였다. 반대로, k 값이 너무 크면, 불필요한 연산이 증가하여 응답 시간이 길어졌다. 이러한 trade-off 를 고려했을 때, k=8 에서 캐시 적중률 99.53%와 평균 응답 시간 3.19 초를 기록하며 두 지표 간 가장 적절한 균형점을 확인할 수 있었다, 이에, 최적의 클러스터 수는 k=8 로 결정하였다.

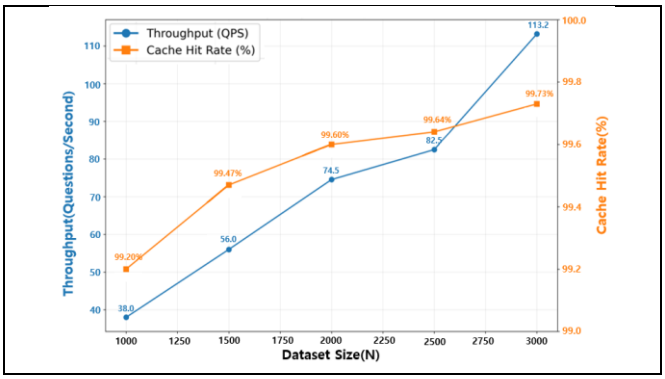


(그림 4) PCA 차원 수(d) 비교

(그림 4)는 차원 축소 효과를 확인하기 위해 PCA 기반 실험을 진행한 결과를 나타낸다. 차원 수가 감소할수록 연산 속도는 빨라졌으나, 정보 손실로 인해 캐시 적중률은 낮아지는 현상이 관찰되었다. 반대로, 차원 수가 지나치게 크면 연산량 증가로 인해 처리 시간이 길어졌다. 이러한 결과를 종합했을 때, 32 차원에서 성능과 효율성의 균형이 가장 안정적으로 확인되었다. 이에, PCA 의 최적 차원 수는 32 로 결정하였다.

4.3 데이터 셋 확장성 분석

제안하는 시스템의 확장성을 검증하기 위해 데이터 셋 크기를 1,000 개에서 3,000 개까지 점진적으로 확대하며 성능을 측정하였다.



(그림 5) 데이터 셋 크기에 따른 캐시 적중률 및 처리량의 변화

(그림 5)는 데이터 셋 크기에 따른 캐시 적중률과 처리량의 변화를 나타낸다. 데이터 셋 크기가 증가함에 따라 전처리 단계에서 수행되는 클러스터링 시간은 다소 증가하는 경향을 보였다. 그러나 실시간 추론 단계의 평균 응답 시간은 0.015 초에서 0.017 초 범위에서 안정적으로 유지되어, 데이터 셋 크기 증가가 실시간 처리 속도에 미치는 영향은 크지 않았다. 처리량(Throughput, QPS) 역시 데이터 셋 크기에 따라 일부 변동은 있었으나, 전반적으로 초당 약 58~67 건 수준을 유지하며 안정적인 성능을 보였다.

특히 주목할 만한 점은 캐시 적중률(Cache Hit Rate)의 변화이다. 데이터 셋이 확장됨에 따라 캐시 적중률은 지속적으로 상승하였으며, 3000 개의 데이터 셋 규모에서는 99.73%에 도달하였다. 이는 캐시를 사전에 구축하려는 전략이 대규모 환경에서도 효과적임을 보여주는 중요한 근거로, 데이터가 많아질수록 오히려 캐시 활용도가 강화됨을 의미한다.

종합적으로, 본 실험은 제안하는 캐싱 기반 시스템이 데이터 셋의 규모가 확장되더라도 평균 응답 시간과 처리량을 안정적으로 유지하고 동시에 캐시 적중률이 향상되는 특성을 보였다. 이러한 결과는 제안하는 시스템이 대규모 데이터 환경에서도 우수한 확장성을 확보하고 있음을 명확하게 보여준다.

5. 결론 및 시사점

본 연구는 **질문의 의미적 유사도를 활용**하여 LLM 의 KV 캐시를 사전 생성 및 재사용하는 새로운 최적화 시스템을 제안한다. 실험 결과, 제안 시스템은 기존 CAG 시스템 대비 추론 속도와 캐시 적중률을 압도적으로 개선하였으며, 특히 질의 반복성이 높은 금융 QA 도메인에서 **실질적인 상용화 가능성**을 입증하였다.

이는 LLM 추론 최적화가 **단순 연산량 축소**를 넘어 **입력 데이터의 의미적 구조**를 활용하는 새로운 접근으로 확장되었음을 보여준다. 더불어, **비지도 학습 기법인 클러스터링**이 LLM 시스템의 성능 개선에 직접 기여할 수 있음을 실험으로 입증한데 의의가 있다.

향후 연구에서는 금융 도메인 내 주식, 펀드, 대출 등과 같은 세부 QA 유형 별로 특화된 **최적의 캐시 구조를 설계**와 **계층적 캐시 관리 시스템** 구축을 통해 시스템의 적응성과 효율성을 높일 예정이다. 아울러, 기존의 속도 및 캐시 적중률 평가 외의 **KV 캐싱에 필요한 메모리 사용량도 함께 평가**하여, 제안된 최적화 시스템의 **종합적인 효용성**을 검증할 것이다.

참고문헌

[1] Chan, Brian J., et al. "Don't do rag: When cache-augmented generation is all you need for knowledge tasks." *Companion Proceedings of the ACM on Web Conference 2025*. (2025).

[2] Tao, Keda, et al. "Plug-and-Play 1. x-Bit KV Cache Quantization for Video Large Language Models." *arXiv preprint arXiv:2503.16257* (2025).

[3] Von Luxburg, Ulrike. "A tutorial on spectral clustering." *Statistics and computing* 17.4 (2007): p.395-416.

[4] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." *arXiv preprint arXiv:1908.10084* (2019).

[5] Shlens, Jonathon. "A tutorial on principal component analysis." *arXiv preprint arXiv:1404.1100* (2014).