RAG 검색 연산의 실행 특성 분석을 통한 최적화 가능성 연구

박기현[1], 이제운[2], 서지원[3]
[1] 한양대학교 컴퓨터소프트웨어학과 석박사통합과정
[2] 한양대학교 컴퓨터소프트웨어학과 학부생
[3] 한양대학교 컴퓨터소프트웨어학과 교수

jelite@hanyang.ac.kr, jewoonlee@hanyang.ac.kr, seojiwon@hanyang.ac.kr

A Study on Optimization Possibilities through Analysis of Execution Characteristics of RAG Retrieval Operations

Gihyun Park[1], Jewoon Lee[2], Jiwon Seo[3]
[1]Dept. of Computer Science, Hanyang University
[2]Dept. of Computer Science, Hanyang University
[3]Dept. of Computer Science, Hanyang University

요 약

본 연구는 Retrieval-Augmented Generation (RAG) 시스템에서 검색 연산의 실행 성능을 관측/분석하고, 추가적인 실행 성능 최적화 가능성을 탐구한다. RAG 는 대규모 언어 모델(LLM)의 한계를 보완하기 위해 외부 데이터베이스에서 관련 정보를 검색하여 모델의 답변 품질을 향상시키는 기술이며, RAG 시스템에서 가장 많이 사용되는 연산은 검색 연산이다. 본 연구에서는 대표적인 벡터 데이터베이스 중 FAISS 와 Chroma 두 개를 선택하여, 여러 실행 조건들의 변화에 따른 지연 시간 변화를 분석하였다. 두 벡터 데이터베이스의 다른 최적화 방식이 대규모 벡터 데이터베이스에서 각각 다른 특성을 가지는데, FAISS 의 경우 데이터베이스의 크기 변화에 덜 민감하고, Chroma 는 특정 구간 안에서는 지연 시간이 증가하다 임계치를 넘어가면 감소하고 다시 증가하는 패턴을 보인다. 본 연구에서는 이에 대한 원인을 분석하여 RAG 시스템의 검색 연산의 더 세밀한 최적화 가능성을 제시한다.

1. 서론

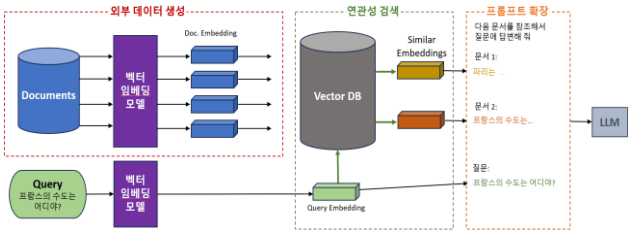
최근 인공지능(AI) 기술의 발전과 함께 대규모 언어 모델(LLM)에 대한 관심이 높아지고 있다. LLM 은 GPT[1], BERT[2], Llama[3] 등 다양한 모델을 통해 자연어 처리 및 데이터 분석 작업에서 핵심적인 역할을 하고 있다. 이러한 모델들은 방대한 양의 데이터를 기반으로 훈련되어 자연어 이해와 생성 능력을 갖추고 있으며, 다양한 산업 분야에서 광범위하게 활용되고 있다. 그러나 LLM 은 학습 이후에 발생하는 새로운 정보는 반영하지 못하며, 이 때문에 환각 답변이나 오래된 정보 제공 등의 문제가 발생할 수 있다. 특히, 이러한 문제는 빠르게 변화하는 정보 환경에서 심각한 제약이 될 수 있다. 이를 해결하기 위해 등장한 Retrieval-Augmented Generation(RAG)[4] 기술은 LLM 이 외부 데이터베이스를 참조하여 보다 정확한 정보를 제공할 수 있도록 한다.

RAG 는 검색 연산을 통해 관련 데이터를 추출하고 이를 LLM 의 프롬프트에 반영함으로써 답변의 품질을 높인다. 특히, 벡터 데이터베이스는 대규모 비정형 데이터를 효율적으로 검색하기 위해 필수적인 구성 요소로 자리 잡고 있다. 벡터 데이터베이스는 벡터 임베딩을 활용해 유사한 정보를 빠르게 검색할 수 있으며, FAISS[5]와 Chroma[6] 같은 시스템이 대표적인 구현체로 사용되고 있다.

본 연구는 RAG 시스템의 성능 최적화를 위해 대표적인 두 벡터 데이터베이스(FAISS, Chroma)의 최적화 방식을 비교 분석하고, 각 데이터베이스의 검색 연산 성능 변화와 그 원인을 탐구하는 것을 목표로 한다. 특히, 검색 연산이 LLM 추론 시스템 전체 성능, 특히 Time to First Token(TTFT)에 큰 영향을 미치는 점에 주목하여 검색 연산의 추가적인 최적화 가능성을 제시하고자 한다.

2. 배경

2.1. RAG (Retrieval Augmented Generation)



(그림 1) RAG 실행 과정

LLM 을 활용할 때의 단점 중 하나는 모델이 학습 완료된 시점 이후에 발생한 새로운 정보는 재학습 없이 답변에 반영되지 못한다는 것이다. 이로 인해 (1) 충분히 답변할 수 있는 근거에 대한 정보가 존재함에도 환각 답변을 생성하거나 (2) 사용자가 기대하는 것보다 일반적이거나 오래된 정보를 제공하는 문제, (3) 신뢰할 수 없는 출처로부터 응답을 생성하는 문제 등이 발생할 수 있다.

RAG 는 앞서 기술한 LLM 의 문제점을 해결하여 출력 문장의 품질을 향상시키는 기술이다. RAG 는 훈련되지 않은 데이터로 이루어진 데이터베이스를 참조하여 LLM 이 더 정확하고 신뢰성 있는 응답을 생성하도록 돕는다. 이러한 방식을 통해 특정 도메인이나 조직의 내부 지식을 참조하여 모델을 재학습하지 않고도 LLM 이 가지고 있는 문장 생성 성능 활용도를 최대화할 수 있다.

RAG 의 실행 과정은 외부 데이터 생성(Create External Data), 관련 정보 검색(Retrieve Relevant Information), LLM 프롬프트 확장(Augment the LLM Prompt) 순서로 이루어지는데, 먼저 학습되지 않은 외부 데이터를 임베딩 모델을 통해 벡터화하고, 변환된 벡터들을 벡터 데이터베이스에 저장한다. 이후 질의문이 입력되면 같은 임베딩 모델을 통해 질의문을 벡터화하여 이를 벡터 데이터베이스에 저장되어 있는 문서 벡터 중 가장 유사한 문서를 검색한다. 이 때, 검색을 위해 수학적인 거리 계산(e.g. 코사인 유사도, 유클리드 거리 등) 또는 기계학습 모델을 활용할 수 있다. 그 이후 검색된 벡터를 다시 문장으로 변환하고 사용자 입력에 적절히 추가하여 LLM 프롬프트를 확장하면 LLM 이 검색된 문서를 참조하여 더욱 정확한 답변이 생성될 수 있다.

2.2 벡터 임베딩

벡터 임베딩[7]은 주제, 단어, 이미지 또는 기타 데이터를 숫자로 표현한 것이다. 벡터 임베딩은 LLM 및 기타 AI 모델에 의해서 생성된다. 각 벡터 임베딩 사이의 거리는 벡터 데이터베이스 또는 벡터 검색 엔진을 통해서 벡터 간의 유사성을 결정할 수 있게 한다.

2.3. 벡터 데이터베이스

벡터 데이터베이스는 이미지, 텍스트, 센서 데이터와 같은 비정형 또는 반정형 데이터를 벡터 임베딩을 통해 벡터로 변환하여 저장하는 데이터베이스이다. 벡터 데이터베이스는 벡터 임베딩을 기반으로 데이터의 검색, 삭제, 삽입 등의 데이터 관리 기능을 수행한다. 본 연구에서는 대표적인 벡터 데이터베이스 구현체인 FAISS 과 Chroma 의 성능을 비교한다.

2.3.1 FAISS

FAISS 는 페이스북 AI 연구팀에서 개발한 벡터 데이터베이스 라이브러리로, 대량의 고차원 벡터에서 유사성 검색을 빠르고 효율적으로 수행할 수 있다. FAISS 는 큰 크기의 벡터를 양자화하여 인덱스를 생성하고, 양자화된 인덱스를 사용하여 입력된 쿼리 벡터와 데이터베이스에 저장된 벡터들 간 유사도 계산을 효율적으로 할 수 있다. 벡터 유사도 계산을 빠르게 하기 위해 FAISS 는 복잡한 벡터 공간을 유사성을 가진 작은 클러스터로 분할하고 원본 데이터 대신 클러스터를 사용하는 방식으로 더 작은 메모리 공간과 계산 유닛으로도 효율적인 데이터베이스 관리를 용이하게 한다.

2.3.2 Chroma

Chroma 는 대용량 문서를 여러 개의 청크로 나누어 검색 성능을 최적화한다. 청크를 나누는 청킹 전략은 크게 두 가지로 나뉘는데, 텍스트의 구조나 의미를 고려하지 않고 일정한 문자 또는 토큰 수로 청크를 나누는 방식(e.g. Recursive Character Text Splitter, Token Text Splitter 등)과, 텍스트의 의미적 유사성을 고려해 청크 크기를 최적화하여 관련 문장들을 묶는 방식(e.g. Cluster Semantic Chunker, LLM Chunker 등)이 있다.. 의미 기반 청킹은 더 정확한 검색을 가능하게 하지만, 문자나 토큰 기반 방식은 단순하고 빠르게 적용된다.

3. 연구 목적 및 필요성

본 연구는 Retrieval-Augmented Generation (RAG) 모델의 Retrieval 단계에서 대표적인 두 벡터 데이터베이스(FAISS, Chroma)에서 다양한 임베딩 모델, 데이터베이스 크기, 검색 결과 벡터 수에 따른 실행 시간 변화를 분석하여 각 요소가 RAG 의 Retrieval 성능에 미치는 영향을 정량적으로 평가하고, 이를 통해 RAG 시스템의 추론 성능을 위한 가능성을 탐색하는 것을 목적으로 한다.

RAG 시스템의 성능 최적화는 산업계의 요구사항에 적합한 주제인데, 특히 기업의 내부 문서 관리, 고객 서비스, 지식 기반 시스템 등 광범위한 영역에서 RAG 의 활용이 증가하는 것을 고려할 때 RAG 시스템에서 가장 빈번하게 수행되는 연산인 검색 연산 최적화를 통하여 LLM 추론 실행을 위한 전체 산업계의 요구 자원 및 에너지 사용량 감소를 기대할 수 있다.

검색 연산은 전체 시스템의 성능, 특히 추론 시간에 지대한 영향을 미치는데, 특히 주목해야 할 점은 검색 연산이 Time To First Token(TTFT), 즉 시스템이 첫 번째 응답 토큰을 생성하는 데 걸리는 시간에 직접적인 영향을 준다는 것인데, 그 이유는 사용자 입

력이 추론 시스템에 도달하고 나서 검색 연산이 완료될 때까지 LLM 의 Prefill 연산이 실행되지 못하고 기다려야 하기 때문이다. 사용자 경험과 시스템의 실시간 대응 능력에 크게 영향을 미치는 TTFT 의 최소화를 통해 RAG 시스템의 실용성과 효율성을 크게 향상시킬 수 있다.

따라서 RAG 검색 연산의 최적화는 단순히 시스템의 속도 향상뿐만 아니라, 전반적인 성능 개선과 사용자 만족도 증대를 위해 반드시 필요하다. 이러한 최적화를 통해 RAG 시스템은 더욱 빠르고 효율적으로 작동할 수 있게 되어, 결과적으로 생성형 AI 기술의 실용성과 적용 범위를 크게 확장할 수 있을 것이다. 본 연구에서는 이러한 배경을 바탕으로 RAG 검색 연산의 최적화를 위한 가능성을 탐구하고자 한다.

4. 실험

4.1 실험 환경 및 실험방법

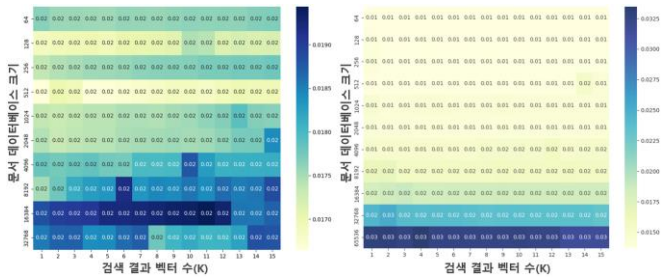
본 논문에서 사용된 하드웨어 환경은 Nvidia V100 GPU 와 Nvidia 2080Ti GPU 로 구성된다. Nvidia V100 GPU 는 PCIe 3.0x16 을 통해 인텔 제온 E5-2698 CPU 가 장착된 시스템에 연결되었으며, 2080Ti GPU 는 PCIe 3.0x16 을 통해 인텔 제온 E5-2620 CPU 가 장착된 시스템에서 사용되었다. “4.2.5 임베딩 모델의 파라미터 크기에 대한 비교” 실험은 Nvidia 2080Ti GPU 에서 실행하였고, 나머지 실험들은 Nvidia V100 GPU 에서 실행되었다. 실험방법은 임베딩 모델, 벡터데이터베이스, 데이터베이스 크기, 반환되는 벡터 수를 변경하며 RAG 검색 연산의 지연 시간을 측정하는 방식으로 진행되었다. 사용된 실험 요소는 <표 1>과 같다.

모델명	벡터 크기	파라미터 수
MiniLM-L6-v2	384	22M
MiniLM-L12-v2	384	118M
all-mpnet-base-v2	768	109M
gte-Qwen2-1.5B-instruct	1536	1.78B

<표 1> 실험에 사용한 모델 구성요소

4.2 결과 및 분석

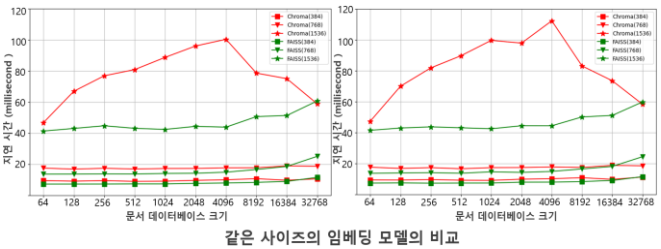
4.2.1 FAISS 와 Chroma 비교



(그림 2) 조건별 지연 시간 비교 (좌 Chroma, 우 FAISS)

(그림 2)는 all-mpnet-base-v2 모델에서 FAISS 와 Chroma 벡터 검색 엔진의 지연 시간을 비교한 히트맵이다. x 축은 반환된 검색 결과 벡터 수, y 축은 문서 데이터베이스 크기를 나타낸다. 각 색상은 진할수록 긴 지연 시간을 나타낸다. Chroma 는 데이터베이스 크기가 증가할수록 지연시간이 증가하는 반면, FAISS 는 일정한 성능을 유지한다.

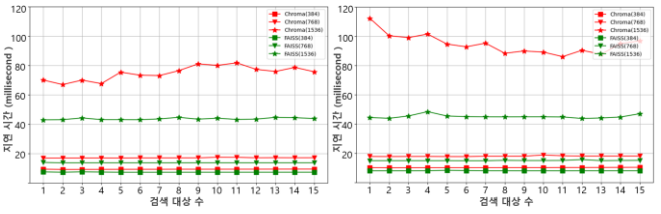
4.2.2 문서 데이터베이스 크기에 대한 비교



(그림 3) 문서 데이터베이스 크기에 따른 지연 시간 변화 (검색 대상 수(K)-좌:1, 우:15) *모델명(벡터크기)

(그림 3)에서는 Chroma, FAISS 두 모델 모두 벡터 차원이 768 보다 작은 경우 문서 데이터베이스 크기가 커져도 지연 시간은 일정하게 유지된다. 반면, 벡터 차원이 1536 이상인 경우 데이터베이스 크기에 따라 지연 시간이 변동한다. 동일한 조건에서 Chroma 와 FAISS 의 차이를 보면, FAISS 가 Chroma 보다 더 낮은 지연 시간을 보여준다.

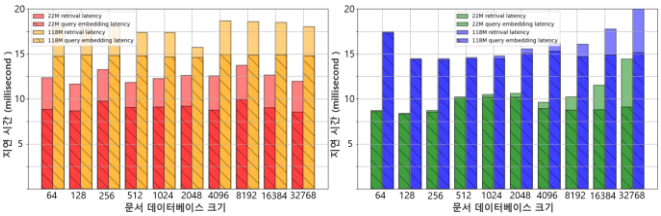
4.2.4 검색 대상 수(K)에 대한 비교



(그림 4) 반환되는 벡터 수(K)에 따른 지연 시간 변화 (벡터크기-좌:128/우:4096) *모델명(벡터크기)

(그림 4)는 Chroma, FAISS 두 모델 모두 벡터 차원이 768 보다 작은 경우 반환되는 벡터 개수가 많아져도 일정한 지연 시간이 다. 반면 벡터 차원이 1536 이상인 경우 반환되는 벡터 개수에 따라 시간이 변동한다. 동일한 조건에서 Chroma 와 FAISS 의 차이를 보면, FAISS 가 Chroma 보다 더 낮은 지연 시간을 보인다.

4.2.5 임베딩 모델의 파라미터 크기에 대한 비교



(그림 5) 모델 크기 별 쿼리 임베딩, 검색속도 실행시간 변화(데이터베이스 종류-좌:Chroma, 우:FAISS)

(그림 5)는 Chroma 와 FAISS 의 임베딩 벡터 크기는 384 로 같지만 파라미터 수가 22M 및 118M 로 다른 MiniLM-L6-v2m, MiniLM-L12-v2 모델에 대한 그래프이다. 빗금 친 부분은 쿼리 임베딩 시간을 의미한다. FAISS 는 Chroma 에 비해 22M 크기의 모델에 대해

더 낮은 지연시간을 보이며, (그림 4)와 비교해보았을 때 RAG 검색 연산은 모델의 파라미터 크기보다 문서 데이터 베이스 크기가 더 큰 영향이 있음을 보여준다.

4.3 Case Study

‘문서 데이터베이스 크기에 따른 지연 시간 변화’그래프를 보면 문서 데이터베이스 크기가 4096 이후로 데이터베이스 크기가 커질수록 지연 시간이 감소하는 경향을 확인할 수 있다. 이는 Chroma 에서 지원하는 청킹(chunking)[8] 전략에 기인한다. Chroma 는 데이터베이스가 클 때, 문서 전체를 검색하지 않고 데이터 청크(일부로 나눈 데이터)를 검색하여 필요한 정보를 효율적으로 찾아낸다. 임베딩 크기가 충분히 클 때는, 각 데이터 청크가 더 많은 정보를 포함하고 있어 청킹 전략의 효율성이 극대화 된다. 그러나 임베딩 크기가 작을 경우, 청크별 정보의 밀도가 낮아지기 때문에 청킹 전략의 이점이 줄어들 수 있다. 이로 인해 Chroma 는 작은 임베딩 크기에서는 청킹 전략을 사용하지 않는다.

반면 FAISS 는 IVF(Inverted file with exact post-verification), HNSW(Hierarchical Navigable Small World graph exploration) 인덱싱 방법[9]을 통해 검색 효율성을 유지할 수 있다. IVF 는데이터를 여러 클러스터로 나누고, 탐색 시 관련 클러스터에서만 검색을 수행하여 불필요한 계산을 줄여 검색 속도를 높이는 방법으로 대규모 데이터셋에 효율적이다. 또한 HNSW 는, 벡터들이 서로 근접한 이웃과 연결된 계층적 그래프를 구축하여 검색 시 이 그래프를 따라 효율적으로 탐색함으로써 빠르게 최근접 이웃을 찾는 방법이다. 이러한 방법을 통해 FAISS 는 Chroma 에 비해 비교적 일찍 낮은 시간으로 탐색을 할 수 있다.

5. 결론

본 연구에서는 Retrieval-Augmented Generation (RAG) 시스템의 핵심 요소인 검색 연산의 최적화 가능성을 다양한 실험을 통해 분석하였다. 특히, 두 가지 벡터 데이터베이스(FAISS 와 Chroma)를 활용하여 각 시스템의 성능을 비교하고, 지연 시간에 미치는 영향을 살펴보았다.

실험 결과, Chroma 는 청킹(chunking) 전략을 통해 대규모 문서 데이터베이스에서도 지연 시간를 크게 줄일 수 있음을 확인하였다. 데이터베이스 크기가 증가할수록 지연 시간이 감소하는 현상은 Chroma 가 문서 전체를 검색하는 대신 청크로 나눈 데이터를 검색하여 효율적으로 필요한 정보를 찾아내기 때문이었다. 반면, 임베딩 크기가 작을 경우 청킹 전략의 효율성이 감소하여 연산 속도에 효율성이 감소함을 알 수 있었다. 반면에 FAISS 는 IVF(Inverted File Index)와 HNSW(Hierarchical Navigable Small World)와 같은 인덱싱 방법을 사용하여 대규모 데이터베이스에서도 일정한 검색 성능을 유지할 수 있었다. 이러한 인덱싱 방법을 통해 FAISS 는 검색 시 불필요한 데이터를 배제하고, 관련성 있는 데이터에만 집중하여 연산 속도를 크게 향상시킨다. 특히 IVF 는 데이터를 여러 클러스터로 분할하여 검색할 때 관련 클러스터에서만 탐색을 진행함으로써, 처리 시간을 줄이는 데 중요한 역할을 한다. HNSW 는 벡터 데이터 간의 연결 구조를 계층적으로 설정하여, 빠르게 가까운 이웃을 탐색할 수 있도록 돕는다. 이러한 기술적 차이를 통해 FAISS 는 대규모 데이터베이스 환경에서도 안정적으로 낮은 지연 시간을 유지할 수 있었다.

결과적으로, FAISS 는 Chroma 에 비해 대규모 데이터셋에서 더 일관된 성능을 제공하며, 특히 임베딩 크기가 커질수록 이러한 차이는 더욱 두드러졌다. 이러한 실험 결과는 RAG 시스템의 검색 연산 최적화를 위한 중요한 방향성을 제시하며, 특정 상황에 따라 FAISS 와 Chroma 각각의 장점을 극대화할 수 있는 전략을 수립할 필요가 있음을 보여준다.

최종적으로, 본 연구는 RAG 시스템의 검색 연산 최적화가 전체 추론 성능에 미치는 중요한 영향을 강조하며, 이를 통해 실시간 응답 능력과 사용자 경험을 향상시킬 수 있는 방안을 제시하였다. 앞으로의 연구에서는 다양한 임베딩 모델과 더 복잡한 검색 구조를 고려한 실험이 필요할 것이며, 이를 통해 더욱 효율적인 RAG 시스템 설계가 가능할 것으로 기대된다.

<사사문구>

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터사업의 연구결과로 수행되었음 "(IITP-No.2024-2021-0-01817,No.RS-2020-II201373, No.2022-0-00498)

참고문헌

[1] Brown, Tom B, "Language models are few-shot learners.", arXiv preprint arXiv:2005.14165, 2020.

[2] Devlin, Jacob, "Bert: Pre-training of deep bidirectional transformers for language understanding.", arXiv preprint arXiv:1810.04805, 2018.

[3] Dubey, Abhimanyu, et al, "The llama 3 herd of models.", arXiv preprint arXiv:2407.21783, 2024.

[4] Lewis, Patrick, et al, "Retrieval-augmented generation for knowledge-intensive nlp tasks.", Advances in Neural Information Processing Systems 33, (2020): 9459-9474.

[5] Douze, Matthijs, et al. "The faiss library." arXiv preprint arXiv:2401.08281, 2024.

[6] Chroma is the open-source AI application database. Batteries included., https://www.trychroma.com/

[7] Almeida, Felipe, and Geraldo Xexéo, "Word embeddings: A survey.", arXiv preprint arXiv:1901.09069, 2019.

[8] Chroma Technical Report, https://research.trychroma.com/evaluating-chunking

[9] Faiss indexes, https://github.com/facebookresearch/faiss/wiki/Faiss-indexes