Journal of KIIT. Vol. 24, No. 8, pp. 69-80, Aug. 31, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.8.69

질의 조건 기반 Retrieval Unit을 적용한 검색 증강 생성 기법

김영아*[1], 정선미*[2], 김현주*[3], 김창근**

A RAG Method based on Query-Conditioned Retrieval Units

Yeong-A Kim*[1], Seon-Mi Jeong*[2], Hyun-Ju Kim*[3], and Chang-Geun Kim**

요 약

본 논문에서는 질의 조건 기반 Retrieval Unit을 적용한 검색 증강 생성(RAG) 기법인 DRAG를 제안한다. 기존 RAG 시스템은 고정 길이 분할이나 슬라이딩 윈도우 방식에 의존하여 질의 의미와 문서 구조를 충분히 반영하지 못하는 한계가 있다. 이에 본 논문에서는 문서 분할을 질의 조건 기반 검색 단위 구성 문제로 재정의하고, 질의 의미와 문서 구조를 함께 고려하여 Retrieval Unit을 동적으로 구성하였다. 실험 결과, DRAG는 Fixed Chunking 대비 Recall@20을 9.0% 향상시키고 중복 문맥 토큰 비율을 14.1% 감소시켰으며, Semantic Chunking 대비 F1 Score를 7.0% 향상시켰다. 이러한 결과는 제안 기법이 검색 재현율과 응답 품질을 개선하는 동시에 생성 단계의 문맥 활용 효율성을 높이는 데 효과적임을 보여준다.

Abstract

This paper proposes DRAG, a Retrieval-Augmented Generation (RAG) method based on query-conditioned Retrieval Units. Existing RAG systems often rely on fixed-size chunking or sliding-window methods, which do not sufficiently reflect query semantics and document structure. DRAG reformulates document segmentation as a query-conditioned retrieval unit construction task and dynamically constructs Retrieval Units by considering both query semantics and document structure. Experimental results show that DRAG improves Recall@20 by 9.0% and reduces the Redundant Context token Ratio (RTR) by 14.1% compared with Fixed Chunking, while improving F1 Score by 7.0% compared with Semantic Chunking. These results demonstrate that the proposed method effectively enhances both retrieval performance and context utilization efficiency.

Keywords

RAG, retrieval unit, query-conditioned retrieval, document chunking, context utilization efficiency

* 경상국립대학교 컴퓨터공학과
- ORCID[1]: https://orcid.org/0009-0009-7246-8423
- ORCID[2]: https://orcid.org/0009-0008-9554-3687
- ORCID[3]: https://orcid.org/0009-0008-8782-4518
** 경상국립대학교 컴퓨터공학과 교수(교신저자)
- ORCID: https://orcid.org/0009-0009-0478-8046

· Received: May 19, 2026, Revised: Jun. 25, 2026, Accepted: Jun. 28, 2026

· Corresponding Author: Chang-Geun Kim
Dept. of Computer Science and Engineering, Gyeongsang National University, Jinju, Korea
Tel.: +82-55-772-3323, Email: cgkim@gnu.ac.kr

Ⅰ. 서 론

최근 대규모 언어 모델(LLMs, Large Language Models)을 활용한 생성 시스템에서는 외부 지식원을 반복적으로 탐색하고, 수집된 정보를 단계적으로 정리하는 심층 연구(DR, Deep Research) 에이전트가 주목받고 있다. DR 에이전트는 동적 추론, 장기 계획, 다중 홉 검색, 반복적 도구 사용을 결합하여 복합 정보 탐색을 수행한다[1].

고정된 검색 결과를 생성 문맥으로 사용하는 검색 증강 생성(RAG, Retrieval-Augmented Generation)과는 다르게 질의 상태를 추적하며 검색, 확인, 정보 정제 과정을 순환적으로 실행한다[2]. 웹 브라우저를 직접 검색하고 웹페이지에서 정보 추출을 인용하여 사실 기반 답변을 생성하는 WebGPT와 생각, 행동, 관찰을 반복하는 ReAct 연구도 LLM이 외부 지식원 및 도구와 상호작용하며 추론과 행동 선택을 결합할 수 있음을 보였다[3][4].

RAG는 LLM의 파라미터 지식만으로 응답을 생성하는 한계를 보완하기 위해 제안된 구조이다[5]. 질의 관련 외부 문서를 검색하고 이를 생성 모델의 입력 문맥으로 활용하여 최신 정보나 특정 도메인 지식을 반영한다. 사전 정의된 청크 단위로 문서를 분할한 뒤 질의 유사도를 기준으로 상위 문서를 선택한다. 이 경우 질의의 복잡도와 정보 요구 수준이 달라져도 검색 절차가 크게 변하지 않으며, 관련이 낮은 문서가 포함되면 응답 품질이 저하될 수 있는 단점을 가진다[6]. 특히 복합 질의나 다중 문서 기반 질의에서는 단일 검색 결과만으로 필요한 근거를 확보하기 어렵다[7].

이러한 단점을 개선하고자 IRCoT는 다단계 질의응답에서 검색과 추론을 교차 수행할 때 성능이 개선될 수 있음을 보였다[8]. Active RAG는 검색 시점과 대상을 능동적으로 결정하는 방식의 필요성을 제시하였고, Self-RAG는 검색 필요성 판단과 생성, 자기 평가를 결합하여 검색 결과의 활용 여부를 조정한다[9][10].

그러나 기존의 반복 검색 기반 또는 추론 검증 기반 RAG 연구들은 주로 검색 단계 자체에 집중하였으며, 검색 결과를 질의 조건에 따라 구조화된 생성 입력으로 재구성하는 과정은 충분히 연구되지 않았다.

검색 문서 수의 증가로 유용한 정보의 포함 가능성은 높아지지만 동시에 관련성이 낮은 노이즈 문서도 증가한다[11][12]. 또한 핵심 근거가 여러 문서에 분산되거나 비효율적으로 배치될 경우 생성 모델은 필요한 정보를 안정적으로 활용하기 어렵다[13]. 따라서 검색 결과의 선택 및 재정렬뿐만 아니라 질의 조건에 따라 검색 문맥 단위를 재구성하는 절차가 필요하다[14][15].

본 연구에서는 고정 청크 기반 RAG에서 발생하는 검색 정밀도 저하와 문맥 분산 문제를 해결하기 위해 질의 조건 기반 Retrieval Unit을 적용한 검색 증강 생성 기법을 제안한다. 제안 기법은 심층 연구 에이전트 기반 검색 증강 생성(DRAG, Deep Research Agent-based Retrieval-Augmented Generation) 프레임워크로 구현된다. 핵심이 되는 Retrieval Unit은 고정된 문서 청크나 단순 순위별 검색 결과 집합이 아니라, 동일 질의에 대해 검색된 후보 문맥 중 의미적으로 연결된 문장, 단락 및 표제 정보를 질의 조건에 맞게 동적으로 통합 및 재구성한 생성 입력 단위로 정의된다.

이러한 구조는 문맥 내 노이즈와 정보 분산을 줄이고 정보 불일치 및 환각(Hallucination) 현상을 완화하며, 결과적으로 검색 효율성과 응답 품질을 개선한다. 실험에서는 기존 방식인 Fixed Chunking, Sliding Window, Semantic Chunking과의 비교를 통해 검색 재현율, 질의응답 성능, 중복 문맥 토큰 비율을 평가한다.

본 논문의 구성은 다음과 같다. 제2장에서 관련 연구를 설명하고, 제3장에서는 DRAG 프레임워크와 Retrieval Unit 구성 방법을 설명한다. 제4장에서는 실험 결과를 분석하고, 제5장에서는 결론 및 향후 연구 과제에 대하여 논의한다.

Ⅱ. 관련 연구

2.1 Retrieval 전략의 한계

RAG는 검색 증강 생성 기술로 외부 데이터 소

스에서 실시간 관련 정보를 검색해 응답 생성을 구현하는 AI 기술이다[16]. RAG 구조는 검색과 생성 두 단계를 핵심으로, 벡터 데이터베이스와 임베딩 모델을 활용한다. 사용자 쿼리를 임베딩 모델을 통해 고차원 벡터로 변환, 의미 유사도를 계산한 후 벡터 데이터베이스의 Top-K 문서를 선택한다. 선택된 문서는 쿼리 벡터와 문서 벡터 간의 코사인 유사도를 기준으로 검색기를 통해 추출된 문맥으로 활용되며 LLM 프롬프트에 결합한 뒤 최종 응답을 생성한다.

그러나 RAG는 환각을 줄이고 성능을 향상시키는 장점을 가진 반면 특정 도메인에서 검색된 문서의 낮은 품질과 모델이 문맥 내 노이즈 간섭에 취약하다는 단점을 가진다. 이러한 문제를 해결하기 위해 검색 수행 방식에 따라 과정을 반복적으로 수행하거나 에이전트 기반으로 확장하여 질의에 따라 검색 전략을 동적으로 조정하는 연구도 제안되고 있다. Iterative Retrieval은 사고, 검색, 관찰 과정을 반복적으로 수행하는 구조이며 Recursive Retrieval은 검색된 문맥을 작은 서브쿼리로 분해한 후 각 검색 결과를 통합하는 구조이다. 또한 Adaptive Retrieval은 검색 수행 여부와 시점을 모델이 동적으로 판단하는 구조로, 생성 중 retrieve, critic 등의 전용 토큰을 사용하여 자기 검색을 수행하는 Self-RAG와 생성 중 토큰 확률이 낮아지는 구간에서 추가 검색을 수행하는 FLARE 등이 대표적이다. 그러나 기존 연구의 대부분은 검색 모델의 성능 향상이나 생성 모델의 개선에 초점을 맞추고 있으며, 검색 결과의 선택 및 활용에만 집중하고 있다. 이에 따라 검색 이후 단계에서 문맥 구성 방식에 대한 추가 연구 필요성이 제기되고 있다.

2.2 문맥 재구성(Context reconstruction)

기존 RAG는 문서 원본을 청크 단위로 고정 분할한 후, 각 청크를 독립적인 검색 단위로 활용한다. 질의와 각 분할된 청크는 동일한 벡터 공간에 임베딩되며, 유사도를 기준으로 관련 청크를 검색하는 밀집 표현 기반의 검색법을 구현하여 관련 정보를 효율적으로 탐색한다. 그러나 독립적인 청크의 검색 단위 처리는 복합 질의 및 다수의 문서를 동시에 처리할 경우 관련 정보가 서로 다른 청크에 분산되어 문맥 단절과 정보 취합이 어려운 단점을 가진다.

국내 RAG 구현 연구에서도 문서 분할과 청크 크기 설정은 검색 성능에 영향을 미치는 주요 요소로 다루어졌으며, 구조화된 법률 문서에서는 Parent-Child Chunking을 통해 문서 구조 정보를 검색 효율에 반영하는 방법이 제안되었다[16][17].

청크 분할은 생성 모델의 입력 토큰 제한과 계산 비용을 결정짓는 중요한 요소이며, 문서를 작은 청크로 나누는 것은 모델의 정보량 조절과 검색 결과에 큰 영향을 미친다. 청크 크기가 증가할수록 검색 단위 안에 많은 문맥이 포함되어 정보 누락 가능성은 줄어들고, 불필요한 정보 포함으로 검색 정밀도와 비례적으로 메모리 사용량과 연산량이 증가한다. 반면 청크의 크기가 감소할수록 노이즈와 연산량은 감소하지만, 여러 청크로 나뉘어 문맥의 단절로 인한 응답 정확도가 낮아진다.

청크를 나누는 방식은 문자 수, 단어 수, 문장 수와 같이 정해진 기준에 따라 문서를 나누는 고정 기준 분할 방식과 문장 경계, 문단 구조, 의미 변화 지점 등을 고려하여 문서를 나누는 의미 기반 분할 방식으로 나뉜다. 고정 기준 분할 방식은 구현이 단순하고 검색 단위를 일정하게 유지하는 장점을 가지지만, 문장의 의미 흐름이나 문단 구조를 충분히 반영하지 못할 경우 핵심 정보가 청크 경계에서 분리될 수 있다. 반면 의미 기반 분할 방식은 문장 경계나 의미 변화 지점을 고려하여 문맥 보존에는 유리하지만, 질의 조건에 따라 필요한 근거를 하나의 생성 입력 단위로 재구성하는 문제까지 해결하지는 못한다. 따라서 청크 분할은 RAG의 검색 효율과 생성 품질에 직접적인 영향을 주는 핵심 과정이지만, 청크 단위 검색만으로는 복합 질의에서 요구되는 문맥 구성을 충분히 보장하기 어렵다. 이에 따라 검색 문맥을 단순한 청크 집합으로 사용하는 것이 아니라, 질의 조건에 따라 의미적으로 연결된 생성 입력 단위로 재구성하는 절차가 필요하다.

2.3 Deep research agents

DRA(Deep Research Agent)는 LLM의 성능 향상을 기반으로 환경을 인식하고 목표에 대한 추론과 행동을 수행하는 능력을 갖춘 시스템이다. DRA는 RAG를 통해 외부 지식 기반 및 실시간 데이터에 접근함으로써 LLM 학습 데이터에 포함되지 않은 최신 정보 및 도메인별 지식을 획득할 수 있다. 이를 통해 에이전트는 더 광범위하고 정확한 정보를 기반으로 추론하고 행동한다. 기존의 RAG가 사용자의 쿼리에 대한 관련 정보를 검색하고 LLM이 응답을 생성하는 상대적으로 수동적인 정보 제공 시스템이었다면, 에이전트형 RAG는 질의 처리에 필요한 외부 지식원과 도구를 선택하는 Routing Agent, 복잡한 질의를 하위 질의로 분해하는 Query Planning Agent, 추론과 행동을 반복하는 ReAct Agent, 계획 및 실행 등의 기능을 통합하는 Plan-and-Execute Agent 등을 포함한다. 이를 통해 복잡한 문제를 자율적으로 처리하는 고도의 문제 해결 시스템으로 진화하고 있다.

그러나 기존 DRA 관련 연구들은 검색을 통해 수집된 문맥을 생성 모델에 적합한 입력 단위로 재구성하는 과정을 충분히 다루지 않고 있다. 특히 복수의 문서에서 수집된 근거가 서로 다른 위치에 분산되어 있을 경우, 이를 질의 조건에 따라 하나의 의미적 문맥 단위로 통합하는 절차가 필수적이다.

본 연구는 이러한 한계를 보완하기 위해 DRA의 동적 탐색 구조와 Retrieval Unit 기반 문맥 재구성 방식을 결합한다. 제안하는 DRAG 프레임워크는 에이전트가 질의 조건에 따라 검색을 수행하는 것에 그치지 않고, 검색된 후보 문맥 중 의미적으로 연관된 문장, 단락, 표제 정보를 하나의 생성 입력 단위로 재구성한다. 이를 통해 기존 에이전트 기반 RAG가 가지는 탐색 유연성을 유지하면서도, 생성 단계에서 활용 가능한 문맥 품질을 향상시키고자 한다.

III. 제안 모델

제안 모델은 심층 연구 에이전트를 기반으로 검색 문맥을 재구성하는 검색 증강 생성 시스템이다. 본 연구에서는 DRAG를 적용하여, 기존 RAG에서 검색된 청크가 그대로 생성 모델에 전달되면서 발생하는 문맥 단절과 정보 분산 문제를 해결하는 것을 목적으로 한다. 검색 문맥의 처리 효율을 높이기 위해 DRAG 에이전트는 질의 처리 모듈과 응답 생성 모듈 사이에 배치되어, 후보 문맥의 관리 및 재검색 처리 과정을 제어하는 중추적인 역할을 수행한다.

3.1 시스템 아키텍처

제안 시스템 아키텍처는 사용자 질의를 처리하는 질의 처리 프로세스, 검색된 문맥을 재구성하는 DRAG 에이전트 프로세스, 그리고 재구성된 문맥을 이용하여 응답을 생성하는 응답 생성 프로세스로 구성된다.

그림 1은 심층 연구 에이전트 기반 검색 증강 생성 구조를 기반으로 설계한 DRAG의 전체 구조이다.

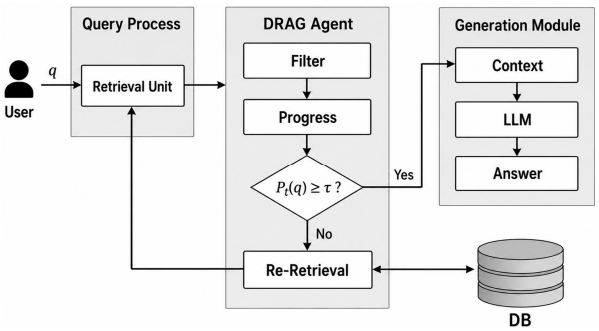

그림 1. 제안하는 DRAG 프레임워크 구조
Fig. 1. Architecture of the proposed DRAG framework

본 시스템의 처리 흐름은 다음과 같다.

Step 1. 질의 처리 프로세스는 사용자 입력 질의와 관련된 후보 문맥을 문서 데이터베이스에서 검색한 후 초기 Retrieval Unit 구성을 실행한다.

Step 2. DRAG 에이전트는 전달된 Retrieval Unit에 대해 필터링 프로세스와 문맥 진행 상태 판단을 실행한다. 필터링 프로세스는 질의와 관련성이 낮은 문서, 반복적으로 포함된 문장, 응답 생성에 사용되지 않는 노이즈 문맥을 제거한다. 이후 문맥 진행 상태 판단 과정은 사용자 질의 조건의 만족도를 평가하기 위해 $P_{t(q)} \geq \tau$ 조건을 사용하며, Retrieval Unit의 질의 충족 정도인 질의 충족 점수 $P_{t(q)}$와 문맥 충분성 임계값 τ를 기준으로 문맥의 충분성을

판단한다.

Step 3. 문맥 진행 상태가 기준값을 만족하지 못하는 경우 DRAG 에이전트는 재검색 프로세스를 실행한다. 재검색 프로세스는 Retrieval Unit에서 부족한 근거 문맥을 보완하기 위해 문서 데이터베이스에 추가 검색을 요청하고, 검색된 문맥은 다시 Retrieval Unit에 포함되며, 필터링 프로세스와 문맥 진행 상태 판단 과정을 반복한다.

Step 4. 문맥 진행 상태가 기준값을 만족하는 경우 응답 생성 프로세스는 전달된 Retrieval Unit을 기반으로 문맥 구성과 최종 응답 생성을 수행한다. 구성된 문맥은 LLM의 입력으로 사용되며, 이때 생성 과정에서는 검색된 전체 청크가 사용되는 것이 아니라 DRAG 에이전트를 통해 정리된 문맥만 사용된다.

제안하는 DRAG 구조는 질의 처리 프로세스와 응답 생성 프로세스 사이에 DRAG 에이전트를 배치한 구조이다. 이 구조에서 검색된 문맥은 필터링, 문맥 진행 상태 판단, 재검색 과정을 거쳐 Retrieval Unit으로 구성된다. 따라서 본 시스템은 기존 RAG의 고정 청크 입력 구조의 한계를 보완하고, 복합 질의 처리에서 필요한 필수 근거 문맥을 생성 모델의 입력으로 효율적으로 구성할 수 있다. 본 논문에서 문맥은 검색된 후보 문서 조각, 문장, 단락 및 표제 정보를 포괄하는 용어로 사용한다.

3.2 질의 조건 기반 Retrieval Unit 구성 모델

DRAG 에이전트는 질의와 관련된 후보 문맥을 평가하고, 관련성 기반 문맥 선별 과정을 통해 생성 입력에 사용될 Retrieval Unit을 구성한다.

기존 RAG에서 사용되는 고정 청크 방식은 문서를 토큰 수, 문자 수, 문장 수와 같은 물리적 기준에 따라 분할하고, 검색된 상위 청크를 생성 모델의 입력으로 전달한다. 그러나 이러한 방식은 질의가 요구하는 의미적 조건과 문맥의 연결 구조를 충분히 반영하기 어렵다. 이에 비해 Retrieval Unit은 질의와 관련된 문장, 단락, 표제 정보를 선별하여 생성 모델에 전달하기 위한 문맥 재구성 단위로 사용된다.

사용자 질의 q에 대해 검색된 후보 문맥 집합을 $C_q = c_1, c_2, \cdots, c_n$이라 할 때, c_i는 후보 청크 안에 포함된 문장, 단락 또는 표제 정보를 의미한다. Retrieval Unit RU(q)는 관련성 필터링, 중복 및 노이즈 제거, 문맥 확장 과정을 거쳐 다음과 같이 구성된다.

$$RU(q) = \{c_i \in C_q \mid Rel(c_i, q) \geq \theta\} \quad (1)$$

식 (1)은 검색된 후보 문맥 C_q 중에서 질의 q와 관련성이 높은 문장, 단락, 표제 정보를 선별하여 Retrieval Unit으로 구성하는 기준을 나타낸다. 본 연구에서 관련성 점수 $Rel(c_i, q)$는 질의와 문맥 요소를 동일한 임베딩 모델로 벡터화한 후, 두 벡터 간 코사인 유사도를 이용하여 계산하였다. 개별 문맥 요소 c_i는 검색된 후보 문맥 안에 포함된 문장, 단락 및 표제 정보를 의미하며, $Rel(c_i, q)$가 관련성 임계값 θ 이상인 경우 해당 문맥 요소를 Retrieval Unit 구성 대상으로 선택하였다. 즉, 식 (1)은 검색된 후보 청크 전체를 사용하는 대신 질의 조건을 만족하는 문맥 요소만을 선별하여 Retrieval Unit을 구성하는 과정을 나타낸다.

RU의 세부 프로세스는 관련성 필터링, 중복 및 노이즈 제거, 문맥 확장으로 구성된다.

첫째, 관련성 필터링 프로세스는 검색된 후보 문맥을 대상으로 문장과 단락의 내용을 질의와 직접 연결되는 핵심 문장이 확인되면 해당 문맥 요소를 RU 구성 대상으로 저장한다. 반대로 질의와 관계가 낮은 설명이나 반복적으로 포함된 문장 또는 응답 생성에 직접 사용되지 않는 일반 문맥은 제거 대상으로 처리한다.

둘째, 중복 및 노이즈 제거 프로세스는 선택된 문맥 요소의 재검사를 통해 같은 의미의 문장과 반복된 문맥을 정리한다. 검색 결과가 늘어나면 필요한 근거 문맥과 함께 연관된 설명도 포함되어 선택된 문맥은 중복 여부와 질의 조건과의 관계를 기준으로 다시 확인한다. 중복으로 판단된 문장은 하나의 문맥 요소로 통합하고, 질의 조건과 거리가 있는 문맥은 RU 구성 대상에서 제외한다.

셋째, 문맥 확장 프로세스는 선택된 대상 문장만

으로 의미 전달이 부족한 경우 실행되며 대상 문장이 포함된 단락 정보와 표제 정보로 문맥의 범위를 보완한 후 최종 RU를 구성한다. 본 연구에서는 이처럼 선택된 문맥 요소와 함께 해당 단락 및 표제 정보를 유지하여 문맥 단절을 줄이는 과정을 구조적 연속성 유지라 정의하며, 이는 문맥 확장 프로세스(아래 알고리즘 1의 Step 3)를 통해 구현된다. 구성된 RU는 DRAG 에이전트의 문맥 진행 상태 판단 과정으로 전달되어 문맥 충분성 평가와 재검색 여부 판단에 사용된다.

Retrieval Unit 구성 프로세스의 핵심 로직은 알고리즘 1과 같다.

알고리즘 1. Retrieval Unit 구성 알고리즘
Algorithm 1. Retrieval Unit construction algorithm

```
Input:
  q, C_q, Θ
Output:
  RU(q)

1. Relevance Filtering
   Select c_i satisfying Rel(c_i,q) ≥ Θ.
2. Deduplication and Noise Removal
   Remove duplicated, irrelevant, and noisy elements.
3. Context Expansion
   Add paragraph and heading information if needed.
4. RU Construction
   Organize selected elements into RU(q).

Goal:
  Generate a query-conditioned Retrieval Unit.
```

알고리즘 1은 사용자 질의 q, 후보 문맥 집합 C_q, 관련성 임계값 Θ를 입력으로 사용하여 질의 조건을 만족하는 문맥 요소를 선별하고 Retrieval Unit을 구성하는 절차를 나타낸다. 먼저 관련성 필터링을 통해 기준값을 만족하는 문맥 요소를 선택하고, 이후 중복 및 노이즈 제거 과정을 통해 불필요한 문맥을 정리한다. 문맥 확장 과정에서는 선택된 문장만으로 의미 전달이 부족한 경우 단락 및 표제 정보를 추가하며, 최종적으로 정리된 문맥 요소를 RU(q)로 구성한다. 이렇게 구성된 Retrieval Unit은 이후 문맥 충분성 판단과 최종 응답 생성을 위한 입력 문맥으로 사용된다.

3.3 문맥 충분성 판단 및 재검색

DRAG 에이전트는 구성된 RU(Retrieval Unit)의 문맥 충분성을 판단하고, 기준을 만족하지 못하는 경우 재검색을 통해 부족한 문맥을 보완한다.

문맥 진행 상태를 판단하기 위해 질의 충족 점수 $P_{t(q)}$를 사용한다. 여기서 t는 재검색 반복 단계를 의미하며, $RU_{t(q)}$는 t번째 단계에서 구성된 Retrieval Unit을 나타낸다. $P_{t(q)}$는 현재 Retrieval Unit이 질의 조건을 만족하는 정도를 나타내며, 문맥 요소의 질의 관련성, 질의 핵심 조건의 포함 여부, 문맥 요소 간 연결성을 종합하여 계산한다. 질의 핵심 조건은 질의에서 요구하는 주요 개체, 조건 및 속성 정보를 의미하며, 문맥 요소 간 연결성은 선택된 문맥이 동일 단락, 인접 문단 또는 동일 표제 아래에서 의미적으로 연결되는지를 기준으로 판단한다.

$P_{t(q)}$가 문맥 충분성 임계값 τ 이상인 경우 현재 Retrieval Unit이 응답 생성에 필요한 충분한 근거 문맥을 확보한 것으로 판단하고 재검색을 종료한다.

문맥 진행 상태 판단 조건은 식 (2)와 같다.
여기서 A_t는 t번째 단계에서 구성된 Retrieval Unit에 대한 처리 결정 결과를 의미한다.

$$A_t = \begin{cases} \text{Accept}, & \text{P}_\text{t}(\text{q}) \geq \tau \\ \text{Re}-\text{Retrieval}, & \text{P}_\text{t}(\text{q}) < \tau \end{cases} \quad (2)$$

식 (2)는 현재 구성된 Retrieval Unit이 응답 생성에 사용될 수 있는지를 판단하는 기준을 나타낸다. $P_{t(q)}$가 임계값 τ 이상이면 A_t는 Accept로 결정되어 현재 Retrieval Unit을 응답 생성에 사용하고, τ보다 작으면 질의 조건을 충분히 만족하지 못한 것으로 판단하여 재검색을 수행한다. 여기서 τ는 문맥 충분성 판단을 위한 임계값으로, 3.2절에서 문맥 요소 선별에 사용한 관련성 임계값 Θ와는 구분된다.

문맥 진행 상태 판단 및 재검색 프로세스의 핵심 절차는 알고리즘 2와 같다.

알고리즘 2는 초기 Retrieval Unit을 입력으로 사용하여 문맥 충분성을 판단하고, 기준을 만족하지 못하는 경우 재검색을 수행하는 절차를 나타낸다. DRAG 에이전트는 현재 RU에 대해 질의 충족 점수 $P_{t(q)}$를 계산하고, $P_{t(q)}$가 임계값 τ 이상이면 해당

RU를 응답 생성에 필요한 문맥으로 사용한다. 반대로 $P_{t(q)}$가 τ보다 작은 경우에는 부족한 문맥 요소를 확인한 후 문서 데이터베이스에서 추가 후보 문맥을 검색하고, 3.2절에서 정의한 관련성 필터링, 중복 및 노이즈 제거, 문맥 확장 과정을 반복하여 RU를 보완한다.

이 과정은 최초 검색 결과만으로 부족한 근거를 반복적으로 보완하고, 복합 질의에서 요구되는 다중 근거를 생성 모델의 입력 문맥으로 구성하기 위한 절차이다. 기준값을 만족한 RU는 응답 생성 모듈로 전달되며, 응답 생성 모듈은 이를 Context로 구성하여 LLM의 입력으로 사용한다.

따라서 제안 구조는 고정 청크 기반 RAG에서 발생하는 근거 부족, 문맥 단절, 노이즈 문맥 유입 문제를 줄이고, 생성 모델에 전달되는 Retrieval Unit의 문맥 구성을 보완한다. 구조적 연속성 유지는 알고리즘 1의 문맥 확장 단계(Step 3)와 관련되며, 선택된 문맥 요소가 포함된 단락 및 표제 정보를 함께 유지하여 문맥 단절을 줄이는 과정을 의미한다. 공동 최적화 과정은 알고리즘 2의 반복 단계에서 질의 관련성과 문서 구조 정보를 통합적으로 반영하여 Retrieval Unit을 구성하는 절차를 의미한다.

알고리즘 2. 문맥 진행 상태 판단 및 재검색 알고리즘
Algorithm 2. Context progress judgment and re-retrieval algorithm

```
Input: Query q, Initial Retrieval Unit RU_0(q),
    Document Database DB, Threshold τ
Output: Final Retrieval Unit RU*(q)

1. Initialize RU_t(q) ← RU_0(q)
2. Compute P_t(q) for RU_t(q)
3. if P_t(q) ≥ τ then
4.       return RU*(q) ← RU_t(q)
5. else
6.       Identify insufficient context elements
7.       Retrieve additional candidate contexts from DB
8.       Apply filtering, deduplication, and context
expansion
9.     Update RU_t(q) with the selected contexts
10. Repeat Steps 2 - 9 until P_t(q) ≥ τ

Goal: Determine whether the RU is sufficient and
refine it when needed.
```

Ⅳ. 실험 및 결과 분석

4.1 실험 환경 및 데이터셋

제안 기법의 유효성을 확인하기 위해 DRAG 프로토타입을 구현하고 기존 RAG 방식과 비교 평가하였다. 실험에는 약 100GB 규모의 문서 코퍼스를 사용하였으며, 문단, 절, 표제 등의 구조 정보를 유지한 상태에서 벡터 인덱스를 구축하였다. 평가 질의는 총 1,000개로 구성하였고, 단순 사실 확인 질의, 다중 조건 질의, 다중 홉 질의, 초기 검색 결과가 불완전한 질의를 각각 250개씩 포함하였다.

각 질의에는 정답 근거 문맥과 기준 응답을 사전에 구축하였다. 정답 근거 문맥은 원문 문서에서 질의에 대한 직접적인 근거를 제공하는 문단 및 관련 문맥으로 구성하였으며, 기준 응답은 해당 근거 문맥을 바탕으로 연구자가 검토하여 작성하였다. 모든 비교 실험은 동일한 문서 집합, 임베딩 모델, 벡터 검색기 및 생성 모델을 사용하여 수행하였다.

실험 환경은 표 1과 같다.

표 1. 실험 환경
Table 1. Experimental environment

Component	Specification
CPU	Intel core i7-8700 3.20 GHz
Memory	20 GB RAM
Storage	512 GB SSD × 8
Operating system	Linux Ubuntu 22.04.1 LTS
LLM	GPT-4o mini (OpenAI API)
Embedding model	BAAI/bge-m3
Vector store	FAISS
Corpus	Document corpus (100GB)
Queries	1,000

평가 과정에서는 검색된 문맥이 정답 근거 문맥을 포함하는지 여부를 확인하고, 생성된 응답과 기준 응답을 비교하였다. Recall@20은 검색 결과 상위 20개 문맥 단위 중 정답 근거 문맥이 포함된 비율로 산출하였다. EM(Exact Match)는 생성 응답과 기준 응답이 완전히 일치한 질의의 비율이며, F1 Score는 생성 응답과 기준 응답 간 토큰 단위 일치도를 정밀도와 재현율의 조화평균으로 계산하였다.

RTR(Redundant Token Ratio)는 생성 모델에 전달된 전체 문맥 토큰 중 의미가 중복되는 토큰이 차지하는 비율로 정의하였다.

4.2 비교 기법 및 성능 평가 결과

제안 기법의 성능을 확인하기 위해 Fixed Chunking, Sliding Window, Semantic Chunking 기반 RAG와 DRAG를 비교하였다. Fixed Chunking은 일정한 길이로 문서를 분할하는 방식이며, Sliding Window는 인접 청크 간 중첩을 통해 문맥 단절을 완화하는 방식이다. Semantic Chunking은 의미 단위로 문서를 분할하여 문맥 응집성을 높인다. 이들 기법과의 비교를 통해 문맥 구성 방식에 따른 검색 성능과 응답 품질의 차이를 평가하였다.

Adaptive RAG 및 Agentic RAG 계열 방법은 검색 계획 수립, 반복 추론, 외부 도구 사용 등 추가적인 구성 요소를 포함하므로, Retrieval Unit 구성 방식 자체의 효과를 분리하여 비교하기 어렵다. 따라서 본 연구에서는 문맥 구성 방식의 차이를 직접 확인할 수 있는 Fixed Chunking, Sliding Window, Semantic Chunking 기반 RAG를 비교 대상으로 선정하였다.

앞 절에서 정의한 평가 지표를 기준으로 성능을 측정하였으며, 결과는 표 2와 같다.

표 2. 성능 평가 결과
Table 2. Performance evaluation results

Method	R@20	EM	F1	RTR(%)
Fixed	0.54	0.38	0.46	34.7
Sliding	0.55	0.42	0.51	29.8
Semantic	0.61	0.48	0.57	25.6
DRAG (proposed)	0.63	0.55	0.64	20.6

표 2에서 DRAG는 Recall@20, Exact Match, F1 Score에서 가장 높은 성능을 보였으며, Redundant Token Ratio는 가장 낮은 값을 기록하였다. Recall@20은 Fixed Chunking의 0.54에서 DRAG의 0.63으로 9.0% 향상되었으며, 비교 기법 중 가장 높은 성능을 보인 Semantic Chunking과 비교해도 2.0% 높았다. F1 Score는 Semantic Chunking의 0.57에서 DRAG의 0.64로 증가하였고, Exact Match 역시 0.55로 가장 높게 나타났다(Fixed Chunking 대비 17.0% 향상). RTR은 Fixed Chunking의 34.7%에서 DRAG의 20.6%로 14.1% 감소하였으며, Semantic Chunking과 비교해도 5.0% 낮은 값을 보였다.

이러한 결과는 DRAG가 검색된 후보 문맥을 질의 조건에 따라 Retrieval Unit 단위로 재구성함으로써 관련 문맥의 포함 비율을 높이고, 문맥 충분성 판단을 통해 부족한 근거 문맥을 보완하였기 때문이다. 또한 질의 관련성이 낮거나 중복된 문맥을 제거함으로써 응답 품질과 문맥 활용 효율을 함께 향상하였다. 이를 통해 DRAG는 기존 문맥 구성 방식보다 높은 검색 재현율과 질의응답 성능을 달성하면서, 동시에 중복 토큰 사용을 줄이는 효과를 확인하였다.

4.3 기존 RAG 시스템과 비교 분석

기존 RAG 시스템은 사용자 질의에 대해 검색기를 실행하고, 검색된 상위 문맥을 생성 모델의 입력으로 전달하는 구조이다. 이 방식은 구현이 단순하고 검색 결과를 빠르게 생성할 수 있으나, 검색된 문맥이 질의 조건을 충분히 만족하는지 판단하는 과정은 포함하지 않는다. 따라서 검색 결과에 중복 문맥이 포함되거나, 질의응답에 필요한 문맥이 일부 누락되는 경우 최종 응답 품질이 저하될 수 있다.

재검색 기반 RAG 시스템은 기본 RAG의 문맥 부족 문제를 완화하기 위해 추가 검색 과정을 포함한다. 재검색 기반 RAG는 최초 검색 결과가 부족한 경우 추가 문맥을 확보할 수 있으나, 검색된 문맥을 의미 단위로 재구성하거나 문맥 간 관계를 정리하는 과정은 제한적이다. 이로 인해 추가 검색을 수행하더라도 유사한 문맥이 반복적으로 포함될 수 있으며, 생성 단계에 전달되는 문맥의 효율성이 낮아질 수 있다.

제안 기법은 검색 결과와 생성 모델 사이에서 후보 문맥을 Retrieval Unit 단위로 재구성하고, 문맥 진행 상태를 판단하는 DRAG 구조이다. 기준을 만족하지 못하는 경우 재검색을 수행하여 부족한 문

맥을 보완함으로써, 단순 검색 결과를 생성 문맥으로 사용하는 기존 RAG의 한계를 보완한다.

기존 RAG 시스템과 제안 기법의 구조적 차이는 표 3과 같다.

표 3. 기존 RAG 방법론과 DRAG의 비교
Table 3. Comparison of existing RAG methods and DRAG

Method	Context recon.	Progress check	Re-ret.
Basic RAG	No	No	No
Re-Ret RAG	No	Partial	Yes
DRAG	Yes	Yes	Yes

표 3에서 기본 RAG는 별도의 문맥 검증 없이 검색 결과를 생성 모델에 전달한다. 이 방식은 단순 사실 확인 질의에서는 적용 가능하지만, 다중 조건 질의나 다중 홉 질의에서는 필요한 문맥이 여러 청크에 분산될 수 있다. 재검색 기반 RAG는 추가 검색을 통해 문맥 부족을 일부 보완하지만, 검색된 문맥 간의 중복이나 관계를 구조적으로 정리하지 못하는 한계가 있다.

반면 DRAG는 검색된 후보 문맥을 Retrieval Unit으로 재구성하고, $P_{t(q)}$를 기준으로 문맥 충분성을 판단한다. 기준을 만족한 RU는 생성 모델의 입력으로 전달되며, 기준을 만족하지 못한 경우에는 재검색을 통해 부족한 문맥을 보완한다. 이를 통해 DRAG는 검색 문맥의 충분성을 높이고 생성 단계에 전달되는 중복 문맥을 줄인다.

또한 제안 기법의 구성 요소별 영향을 분석하기 위해 구성 요소 제거 실험을 수행하였다. 본 실험에서는 전체 DRAG 모델에서 알고리즘 1과 알고리즘 2의 주요 처리 과정을 각각 비활성화한 후 Recall@20의 변화를 비교하였다. Retrieval Unit 구성 모듈 제거는 알고리즘 1의 관련성 필터링, 중복 및 노이즈 제거, 문맥 확장을 제외하고 상위 청크를 그대로 사용한 조건이다.

구조적 연속성 유지 제거는 알고리즘 1의 문맥 확장 단계(Step 3)를 제외한 조건이며, 공동 최적화 제거는 알고리즘 2의 문맥 충분성 판단 및 재검색 반복 과정을 제외하고 초기 $RU_{0(q)}$를 생성 입력으로 사용한 조건이다. 실험 결과는 그림 2와 같다.

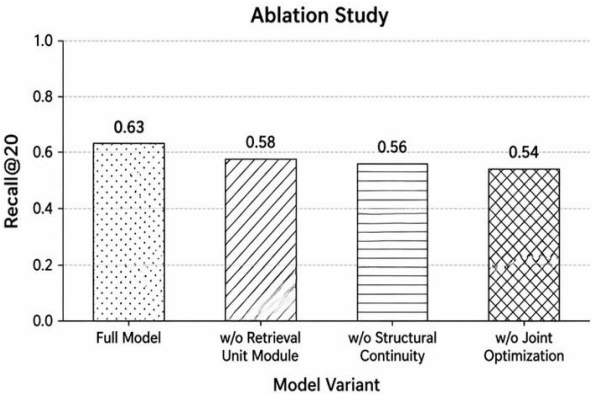

그림 2. Ablation study 결과
Fig. 2. Ablation study results

그림 2에서 전체 모델은 Recall@20 기준으로 가장 높은 성능을 보였다. Retrieval Unit 구성 모듈을 제거한 경우 검색된 문맥을 질의 조건에 따라 재구성하지 못하므로 성능이 감소하였다. 구조적 연속성 유지 과정을 제거한 경우에는 서로 다른 문맥 요소 간 연결성이 약화되어 다중 홉 질의에서 필요한 문맥을 충분히 확보하기 어려웠다. 공동 최적화 과정을 제거한 경우에는 검색 문맥 구성과 재검색 판단이 분리되어 전체 성능이 가장 크게 감소하였다.

이 결과는 제안 기법의 성능 향상이 단일 검색 횟수 증가에 의한 것이 아니라, Retrieval Unit 구성, 문맥 진행 상태 판단, 재검색 제어가 함께 작용한 결과임을 보여준다. 기존 RAG 시스템은 검색 결과를 생성 모델에 직접 전달하는 구조에 가깝지만, DRAG는 검색 결과와 생성 모델 사이에서 문맥을 재구성하고 검증하는 중간 처리 계층을 포함한다. 따라서 제안 기법은 기존 RAG 시스템보다 검색 문맥의 충분성과 응답 생성의 안정성을 높일 수 있다.

4.4 오류 분석 및 한계

제안 기법은 검색된 후보 문맥을 Retrieval Unit 단위로 재구성하고, 문맥 진행 상태에 따라 재검색 여부를 결정한다. 앞선 실험에서 DRAG는 기존 문맥 구성 방식보다 높은 검색 성능과 질의응답 성능을 보였으며, 중복 문맥 토큰 비율도 낮게 나타났다. 그러나 동일한 수준의 개선이 모든 질의에서 나타난 것은 아니며, 검색 문맥의 구성 과정이나 재검색 판단 과정에서 일부 질의의 한계가 확인되었다.

오류가 주로 발생한 경우는 질의 조건이 명확하지 않은 경우였다. 사용자의 질의가 특정 대상이나 비교 기준을 포함하지 않으면, 검색 단계에서 서로 유사한 후보 문맥이 함께 반환된다. 이 경우 Retrieval Unit을 구성하더라도 질의 조건을 충족하는 문맥과 단순히 관련성이 높은 문맥을 구분하기 어렵고, 문맥 진행 상태 판단 과정에서도 충분한 문맥이 확보되었는지 명확하게 판단하기 어렵다.

즉, 재검색을 수행하더라도 동일한 범위의 문맥이 반복적으로 포함되는 경우가 발생할 수 있다. 정답 근거가 문서 집합에 포함되지 않은 경우에도 오류가 발생할 수 있다. DRAG는 검색된 문맥을 재구성하고 부족한 문맥을 보완하는 구조이므로, 문서 집합 내부에 필요한 근거가 존재해야 한다. 그러나 정답 문맥이 검색 대상 문서에 포함되어 있지 않으면, 재검색을 수행하더라도 충분한 Retrieval Unit을 구성하기 어렵다. 이 경우 생성 모델은 제한된 문맥을 기반으로 응답을 생성하게 되며, 응답의 충실도가 낮아질 수 있다.

다중 홉 질의에서는 문맥 간 연결 정보가 부족할 경우 일부 오류가 확인되었다. 여러 문맥을 함께 사용하더라도 중간 근거가 누락되거나 문맥 간 연결성이 약하면 Retrieval Unit이 질의 조건을 충분히 반영하지 못한다. 이 경우 응답은 검색 문맥에 근거하여 생성되지만, 질의가 요구한 전체 조건 중 일부만 포함하는 형태로 나타날 수 있다.

또한 본 연구에서는 평균 성능 비교를 중심으로 제안 기법의 효과를 분석하였으며, DRAG와 비교 기법 간 성능 차이에 대한 통계적 유의성 검정은 충분히 수행하지 못하였다. 표 2의 성능 차이가 우연에 의한 것이 아님을 보다 엄밀하게 입증하기 위해서는 검정 방법과 결과를 구체적으로 보고하는 절차가 추가로 필요하다.

DRAG는 검색 문맥의 충분성과 문맥 활용 효율성을 개선하는 데 효과가 있었으나, 질의 모호성, 정답 문맥 부재, 문맥 간 연결 부족과 같은 상황에서는 한계가 확인되었다. 따라서 향후 연구에서는 질의 조건을 명확화하는 과정과 문맥 간 관계를 판단하는 과정을 추가하여 다양한 질의 유형에서의 적용 가능성을 검토할 필요가 있다.

Ⅴ. 결론 및 향후 과제

본 논문에서는 기존 RAG 시스템에서 검색 문맥이 단순 청크 단위로 구성되는 한계를 분석하고, 이를 해결하기 위해 질의 조건 기반 Retrieval Unit 구성 방법을 제안하였다. 기존의 고정 길이 분할 방식이나 슬라이딩 윈도우 기반 방식은 질의 의미를 충분히 반영하지 못하여 관련 정보가 여러 문서 조각으로 분산되거나 중복 문맥이 포함되는 문제가 존재하였다.

이에 본 연구에서는 문서 분할을 단순한 전처리 과정이 아니라 질의 조건에 따라 검색 단위를 재구성하는 문제로 정의하였다. 제안 기법은 검색 결과와 생성 모델 사이에 배치되어 검색된 후보 문맥을 Retrieval Unit 단위로 구성하고, 문맥 진행 상태를 판단하여 부족한 문맥을 재검색하는 구조로 설계되었다. 이를 통해 질의와 관련된 문맥을 응집된 형태로 구성하고, 생성 단계에 전달되는 중복 문맥을 줄일 수 있도록 하였다. 실험 결과, DRAG는 Fixed Chunking 대비 Recall@20을 9.0% 향상시키고 중복 문맥 토큰 비율을 14.1% 감소시켰으며, Semantic Chunking 대비 F1 Score를 7.0% 향상시켰다.

이러한 결과는 질의 조건 기반 Retrieval Unit 구성이 검색 재현율과 질의응답 성능을 개선하면서도 생성 단계에서 사용되는 문맥 토큰의 중복을 줄이는 데 효과적임을 보여준다.

특히 본 연구는 RAG 시스템에서 문서 분할 문제를 질의 조건 기반 검색 단위 구성 문제로 확장하였다는 점에서 의의를 가진다. 이는 단순한 청킹 기법의 개선을 넘어, 검색된 문맥을 생성 모델에 전달하기 전에 질의 중심으로 재구성하고 검증하는 과정이 중요함을 보여준다.

향후 연구에서는 비교 기법 간 성능 차이에 대한 통계적 유의성 검정을 보다 엄밀하게 수행하고, 제안 기법의 검색 성능 향상에 수반되는 검색 시간 및 연산 비용을 함께 분석하여 성능과 검색 효율성 간의 trade-off를 정량적으로 검증할 필요가 있다. 또한 질의 조건을 명확화하는 과정과 문맥 간 관계를 판단하는 과정을 추가하여 다중 홉 질의에서의 응답 안정성을 높이고, 다양한 도메인 문서 환경에서

제안 기법의 성능을 검증할 예정이다. 아울러 문맥 진행 상태 판단 기준을 질의 유형에 따라 동적으로 조정하는 방식으로 확장할 필요가 있다.

References

[1] Y. Huang, Y. Chen, H. Zhang, K. Li, H. Zhou, M. Fang, L. Yang, X. Li, L. Shang, S. Xu, J. Hao, K. Shao, and J. Wang, "Deep Research Agents: A Systematic Examination and Roadmap", arXiv preprint arXiv:2506.18096, Jun. 2025. https://doi.org/10.48550/arXiv.2506.18096.

[2] A. Singh, A. Ehtesham, S. Kumar, T. T. Khoei, and A. V. Vasilakos, "Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG", arXiv preprint arXiv:2501.09136, Jan. 2025. https://doi.org/10.48550/arXiv.2501.09136.

[3] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman, "WebGPT: Browser-assisted Question-answering with Human Feedback", arXiv preprint, arXiv:2112.09332, Dec. 2021. https://doi.org/10.48550/arXiv.2112.09332.

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models", Proc. International Conference on Learning Representations, Kigali, Rwanda, pp. 1-33, May 2023. https://doi.org/10.48550/arXiv.2210.03629.

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", Advances in Neural Information Processing Systems, Vol. 33, pp. 9459-9474, Dec. 2020. https://doi.org/10.48550/arXiv.2005.11401.

[6] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey", arXiv preprint arXiv:2312.10997, Dec. 2023. https://doi.org/10.48550/arXiv.2312.10997.

[7] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering", Proc. 2020 Conf. on Empirical Methods in Natural Language Processing, Online, pp. 6769-6781, Nov. 2020. https://doi.org/10.18653/v1/2020.emnlp-main.550.

[8] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions", Proc. 61st Annual Meeting of the Association for Computational Linguistics, Toronto, Canada, pp. 10014-10037, Jul. 2023. https://doi.org/10.18653/v1/2023.acl-long.557.

[9] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active Retrieval Augmented Generation", Proc. 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 7969-7992, Dec. 2023. https://doi.org/10.18653/v1/2023.emnlp-main.495.

[10] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection", Proc. International Conference on Learning Representations, Vienna, Austria, pp. 1-30, May 2024. https://doi.org/10.48550/arXiv.2310.11511.

[11] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "RAFT: Adapting Language Model to Domain Specific RAG", Proc. Conference on Language Modeling, Philadelphia, Pennsylvania, USA, pp. 1-14, Mar. 2024. https://doi.org/10.48550/arXiv.2403.10131.

[12] S. Yan, J. Gu, Y. Zhu, and Z. Ling, "Corrective Retrieval Augmented Generation", arXiv preprint arXiv:2401.15884, Jan. 2024. https://doi.org/10.48

550/arXiv.2401.15884.

[13] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval", Proc. International Conference on Learning Representations, Vienna, Austria, pp. 1-18, May 2024. https://doi.org/10.48550/arXiv.2401.18059.

[14] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization", arXiv preprint arXiv:2404.16130, Apr. 2024. https://doi.org/10.48550/arXiv.2404.16130.

[15] G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering", Proc. 16th Conference of the European Chapter of the Association for Computational Linguistics, Online, pp. 874-880, Apr. 2021. https://doi.org/10.18653/v1/2021.eacl-main.74.

[16] E. B. Lee and H. Bae, "A Survey on Recent Research Trends in Retrieval-Augmented Generation (RAG)", The Transactions of the Korea Information Processing Society, Vol. 13, No. 9, pp. 429-436, Sep. 2024. https://doi.org/10.3745/TKIPS.2024.13.9.429.

[17] J. S. Lee, J. M. Lee, and J. H. Yoo, "A Retrieval-Augmented Generation-based LLM Chatbot for Student Q&A in a University Learning Management System", Journal of Broadcast Engineering, Vol. 29, No. 5, pp. 581-595, 2024.

저자소개

김 영 아 (Yeong-A Kim)

2018년 2월 : 경상국립대학교 대학원 IT융합공학과(공학석사)
2024년 8월 : 경상국립대학교 대학원 컴퓨터메카트로닉스공학과 (박사수료)
2017년 9월 ~ 현재 : (주)엔코아 HRD 본부 연구원
관심분야 : LLM, RAG, AI Agent

정 선 미 (Seon-Mi Jeong)

2008년 2월 : 경상국립대학교 컴퓨터과학과(공학사)
2014년 2월 : 경상국립대학교 대학원 IT융합공학과(공학석사)
2017년 2월 : 경상국립대학교 대학원 컴퓨터메카트로닉스공학과 (박사수료)
관심분야 : LLM, RAG, AI Agent

김 현 주 (Hyun-Ju Kim)

1988년 2월 : 경상대학교 컴퓨터과학과(공학사)
1990년 2월 : 숭실대학교 대학원 컴퓨터학과(공학석사)
2000년 2월 : 숭실대학교 대학원 컴퓨터학과(공학박사)
2002년 3월 ~ 현재 : 경상국립대학교 컴퓨터공학과 교수
관심분야 : LLM, RAG, AI Agent, 네트워킹, 로봇, ICT

김 창 근 (Chang-Geun Kim)

1985년 2월 : 경상대학교 컴퓨터과학과(공학사)
1991년 2월 : 경남대학교 대학원 컴퓨터공학과(공학석사)
1999년 2월 : 경남대학교 대학원 컴퓨터공학과(공학박사)
1995년 3월 ~ 현재 : 경상국립대학교 컴퓨터공학과 교수
관심분야 : LLM, RAG, AI Agent, 네트워킹, 로봇, ICT