| 정형 데이터 기반 RAG 시스템의 검색 및 생성 최적화 기법  Implementing Autonomous Driving Scenarios with the CARLA simulator  요 약  대규모 언어 모델의 환각 현상을 방지하기 위해 다양한 방법이 제시되고 있다. 그 중 검색 증강 생성 시스템(RAG)은 외부 정보 검색을 통해 생성 모델이 질문과 관련되면서도 신뢰성 및 일관성있는 대답을 할 수 있도록 돕는 시스템이다. 본 문서는 정형화된 API 응답 데이터를 기반으로 고성능 RAG(검색 증강 생성) 챗봇을 구축하는 과정에서 적용된 데이터 전처리 및 아키텍처 최적화 전략을 기술한다. RAG 시스템의 성능이 검색(Retrieval)의 정확성과 생성(Generation)의 충실도에 의해 결정된다는 전제하에, 본 연구에서는 두 요소를 독립적으로 최적화하는 방안을 제안하고 구현하였다.  이를 위해, 검색 단계에서는 정형 데이터를 자연어 설명문으로 가공하여 임베딩함으로써 의미 검색의 정확도를 극대화하였다. 반면 생성 단계에서는 검색된 문서의 원본 데이터를 모두 활용하여 동적으로 컨텍스트를 재구성함으로써 정보 손실 없는 답변 생성을 유도하였다. 또한, 데이터의 의미적 완결성을 고려하여 의도적으로 청킹(Chunking)을 배제하였으며, LLM의 'Lost in the Middle' 현상을 완화하기 위해 프롬프트 구조 최적화 및 Re-ranker 도입 가능성을 탐색하였다. 본 연구는 정형 데이터 기반 RAG 시스템 설계에 있어 데이터의 특성을 고려한 맞춤형 아키텍처가 필수적임을 제안한다.  ABSTRACT  Various methods are being proposed to prevent the hallucination phenomenon in large language models. Among them, the Retrieval-Augmented Generation (RAG) system assists a generation model in providing relevant, reliable, and consistent answers by retrieving external information. This paper describes the data preprocessing and architectural optimization strategies applied in the process of building a high-performance RAG chatbot based on structured API response data.  Operating on the premise that the performance of a RAG system is determined by the accuracy of its retrieval and the fidelity of its generation, this study proposes and implements a strategy to optimize these two components independently.  To achieve this, in the retrieval stage, we maximized the accuracy of semantic search by processing structured data into natural language descriptions for embedding. Conversely, in the generation stage, we prompted information-lossless answer generation by dynamically reconstructing the context using all the original data from the retrieved documents. Furthermore, we intentionally omitted chunking to preserve the semantic integrity of the data and explored prompt structure optimization and the potential introduction of a re-ranker to mitigate the "Lost in the Middle" phenomenon in LLMs. This research suggests that a customized architecture, which considers the specific characteristics of the data, is essential for designing RAG systems based on structured data.  **키워드 :** RAG, 데이터 전처리, 정형 데이터, 벡터 검색, Lost in the Middle, Langchain  **Key Words :** RAG, data pre-processing, structured data, vector search, Lost in the Middle, Langchain |
| --- |
|  |

Ⅰ. 서 론

인공지능 기술의 발전에 따라 현대 사회에서는 자동화 시스템이 증가하고 있다. 대표적인 자동화 시스템 중 하나인 자율주행 시스템은 운전자의 부담을 줄여주는 반면, 안전 문제와 직결되기 때문에 실제 주행 전 사전 점검이 필수적이다. 이러한 사전 점검은 실제 환경에서도 진행할 수 있지만, 원하는 기후 환경을 연출하거나 강제로 사고 상황을 연출하는 데 있어서는 한계점이 존재한다. 이때 실제 환경에서는 연출하기 어려운 상황 또는 배경 조건을 연출하여 사전 점검을 할 수 있도록 가상의 환경을 조성해주는 프로그램이 자율주행 시뮬레이터이다. 자율주행 시뮬레이터를 이용함으로써 현실과 최대한 유사한 환경 및 상황을 구현할 수 있으며, 실차 테스트에서 하지 못하는 실험 등을 수행하고 평가할 수 있다[1]. 능 연구용 플랫폼으로 적합한 시뮬레이터라 할 수 있다.



그림 1. AirSim 시뮬레이터 실행 화면[2]

Fig. 1. AirSim simulator launch screen[2]

2.2. 벡터 임베딩

시맨틱 검색에 유리한 형태 : 지원기간 : YYMMDD

Simulink

2.3. Lost in the Middle

세번째로 CARLA 시뮬레이터는 오픈소스 자율주행 시뮬레이해 부여한다.

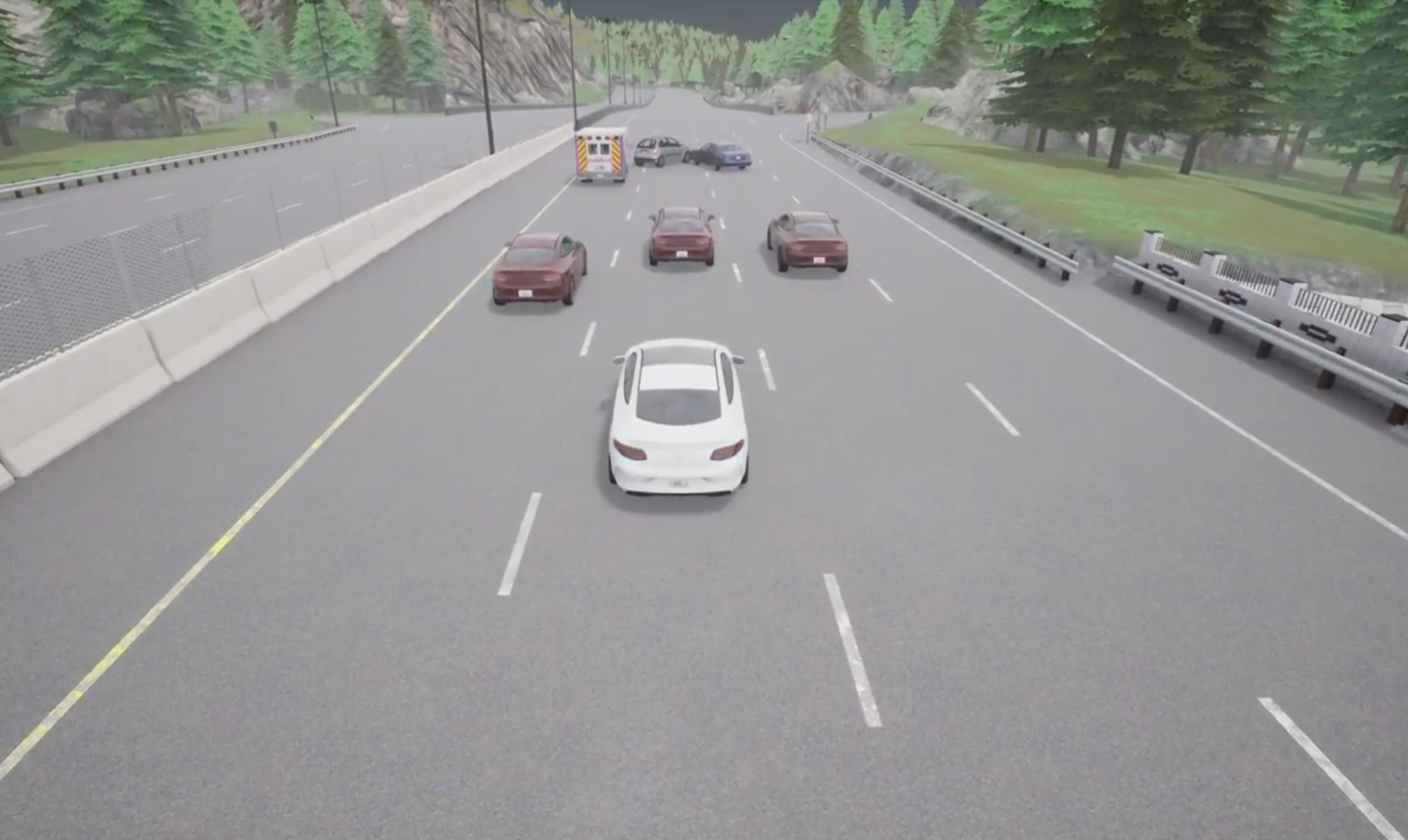


그림 3. CARLA 시뮬레이터 실행 화면

Fig. 3. CARLA simulator launch screen

2.4. Re-Ranker

세 개의 자 구구현하였다.

Ⅲ. 시스템 기본 구성 및 설계 철학

3.1

CA 여기에는 내가 전략을 달리한 부분을 제외한 나머지 공통적인 부분의 처리를 적자.

는 Python API 모듈을 통해 서버에 명령할 수 있다.

3.2 actor 및 blueprint

시뮬레이션 내에서 역할을 수행하는 모든 것들은 CARLA 시뮬레이정 식별자를 갖는다. 가령 차량용 blueprint의 경우 식별자가 ‘vehicle’로 시작하며, ‘.’(온점)을 구분자로 하여 제조사와 차량 모델명으로 구성된다. 센서의 경우 마찬가지로 ‘sensor’로 시작하며 센서의 종류와 기능명으로 구성된다.

3.3 actor 생성

CARLA 시뮬레이터 내에 차량을 생성하고자 한다면, 먼저 차량용 blueprint를 생성해야 한다. 후술할 사용 예시에서는 ‘mercedes.coupe\_2020’ 차량 모델의 blueprint를 사용하였다. 앞서 언급한 것처럼 각 blueprint는 역할에 맞는 적절한 속성을 갖고 있다. 또한 이중 일부 속성의 경우 수정할 수 있다. 사용 예시에서는 blueprint의 ‘role\_name’ 속성을 ‘hero’로 설정하였는데 이는 CARLA 시뮬레이터의 일부 함수들이 ‘role\_name’ 속성을 식별자로 사용하여 작동하기 때문이다. 또한 시뮬레이션 시, 주요 차량의 식별을 용이하게 하기 위해 차량의 색상을 흰색으로 설정하였다.

차량의 기반이 될 blueprint 설정을 완료하였다면, 시뮬레이션 맵에서 차량이 생성될 위치와 회전(rotation) 정보를로 인해 미끄러지는 현상이 발생할 수 있다. 이러한 경우 지면에 차량이 닿은 직후 후진 기어를 적용해주었다 풀어주는 것으로 미끄러지는 현상을 해결할 수 있다.

3.4 actor 제어 : controller

CARLA 시뮬레이터의 actor 제어는 각 actor에 적합한 controller의 생성과 이를 적용하는 것으로 수행된다. 차량 움직임 제어의 경우 vehicle controller를 사용할 수 있다.

표 1. 차량 제어 특성

Table 1. vehicle controller properties

| Properties | Definition |
| --- | --- |
| throttle | Scalar value to control the vehicle's throttle |
| steer | Scalar value that controls the steering of the vehicle. |
| brake | Scalar value to control the vehicle's brakes |
| hand\_brake | Properties that determine whether a vehicle's handbrake is enabled or disabled |
| reverse | Attributes that determine whether a vehicle is in reverse |
| manual\_gear\_shift | properties that determines whether the vehicle's gear should be changed manually |
| gear | Vehicle's gear state properties |

Vehicle controller어가 이루어진다. 이때, 각 조명은 토글 형식이 아니므로 조명을 끄기 위해서는 조명 종료 속성을 적용해야 한다.

Ⅳ. Vehicle controller 기반 주행 시나리오 구현

본 장에서는 실제 사용 예시와 더불어 각 예시가 어떤 목적으로 사용되었으며 어떠한 결과를 내는지, 사용법은 어떠한 지에 대해 상세히 기술한다.

4.1 Usage : lane change

Lane change 시나리오는 주행 중 ego vehicle이 차선을 변경하고자 할 때 주변 차량에게 의도를 전달하여 양보 행동을 유도하고 차선을 변경하는 시나리오이다. Ego vehicle은 자율 주행 시뮬레이션 분야에서 분석의 기준이 되는 차량을 의미한다. 본 장에서는 ego vehicle을 중심으로, 각 예시가 어떤 목적으로 연출되었으며 어떠한 결과를 내는지 기술한다. 동시에 실제 사용 예시와 사용법을 병기한다.

Ⅴ. 검색용 데이터와 생성용 컨텍스트의 분리 및 최적화

첫 번째 방식의 한계를 극복하기 위해, RAG의 검색(Retrieval)과 생성(Generation) 단계를 분리하여 각각에 최적화된 데이터를 공급하는 아키텍처를 설계하였다.

5.1. 검색 단계 최적화

검색 단계의 목표는 사용자의 어떤 질문에도 가장 높은 관련성을 가진 ‘정책’을 정확하게 찾는 것이다. 이를 위하여 다음과 같은 전략을 사용하였다.

5.1.1. 의미론적 문서화 (Semantic Documentarization)

3장에서 제시한 바와 같이, 본 프로젝트에서 사용한 데이터는 정형 데이터이다. 시맨틱 검색의 작동 방식에 근거하여, 정형 데이터를 그대로 임베딩하는 대신 사용자의 질문을 예측하고, 이에 대답하기 위해 필요한 정보 컬럼들을 선별하였다. 이 정보들을 시맨틱 검색에 유리한 형태( 자연스러운 설명문)로 가공하여 단일 ‘document’ 컬럼으로 제작하였다. 이 컬럼은 임베딩 시 page\_content로 설정되어 오직 시맨틱 검색의 정확도를 높이는 데 사용된다.

표. NN 의미론적 문서화 예시

이러한 의미론적 문서화를 통해, ‘기본계획차수’나 ‘정책등록자명’과 같이 검색 과정에 불필요하거나 방해가 되는 정보들을 제거하고 풍부한 문맥 정보를 제공하여 검색의 품질의 향상을 도모하였다.

의미론적 문서화가 장점만 있는 것은 아니다. 기본적으로 전처리 로직의 복잡성이 증가한다. 또한 문서화를 위한 정보 컬럼을 선별하는 과정에서 주관이 개입될 위험이 있으며, 이는 결과적으로 정보 편향으로 이어질 수 있다. 뿐만 아니라 검색 성능을 위해 의도적으로 제외한 정보(e.g., 신청 URL)는 page\_content에 없으므로 이 자체만으로는 완전한 답변을 생성할 수 없다는 명백한 한계가 존재한다.

예를 들어 신청 URL은 사용자에게 제공할 중요한 정보이지만, 시맨틱 검색의 성능 향상을 위해서는 자연어 변환 컬럼에서 제외할 수 밖에 없다. (URL을 제외한 이유는 청킹 전략과도 관련이 있다. 하나의 벡터는 정책의 모든 것을 대표해야한다. 이미 정책 제목과 설명, 기간 등의 핵심 정보를 담고 있는 상황에서 URL을 포함하면 ‘정책을 대표하는’ 벡터의 의미가 희석될 것이라고 판단하였다.)

5.1.2 데이터 특성에 기반한 ‘청킹 배제’ 전략

일반적인 RAG 파이프라인과 달리, 본 프로젝트에서는 의도적으로 청킹을 배제하였다. 청킹의 주 목적은 컨텍스트 윈도우 사이즈 극복, 노이즈 감소, 정보의 밀도 조정 등은 이미 다른 방식으로 해결되었기 때문이다.

* 컨텍스트 윈도우 : ‘document’ 컬럼의 평균 토큰 수는 임베딩 모델의 한계를 초과하지 않는다.
* 노이즈 감소 : 앞선 ‘의미론적 문서화’가 검색에 불필요한 노이즈를 제거하는 ‘논리적 청킹’의 역할을 수행했다.

표. NN. 평균 토큰 수, 1000을 초과하는 문서 수, 2000을 초과하는 문서 수.. .

청킹을 하지 않은 가장 큰 이유는, 데이터의 의미적 완결성을 지키기 위함이다. 하나의 정책 문서는 그 정책에 대한 모든 정보이자 완결된 의미 단위인데, 이를 청킹할 경우 정보가 파편화되어 검색 정확도가 저하될 것이라 판단하였다. 즉, 정책이라는 큰 틀 안에 있어야 정책의 설명, 주관 기관, 사업 기간 등의 정보가 완전한 의미를 가진다는 것이다..

마찬가지로 청킹 배제가 또한 잠재적 트레이드 오프를 갖는다. 정보의 밀도가 조절되었고, 필수적인 정보들만 포함한다 한들 기본적으로 청킹된 문서보다 포함하고 있는 정보가 많다. 따라서 벡터 에서 문서 내 구체적인 정보나 고유 의미가 희석될 수 밖에 없다.

5.2 생성 단계 최적화

생성 단계의 목표는 검색된 정책에 대해, 정보 손실 없이 가장 정확하고 풍부한 답변을 사용자에게 제공하는 것이다.

5.2.1. 동적 컨텍스트 재구성(Dynamic Context Re-generation)

검색용으로 선별 및 가공된 page\_content를 컨텍스트로 제공할 경우, LLM이 특정 질문에 대해서 대답할 수 없거나 할루시네이션이 발생할 가능성이 있다. 이러한 정보 부족 문제를 해결하기 위해 다음과 같은 동적 재구성 전략을 사용하였다.

1. 데이터 저장: 벡터 DB 저장 시, 검색용 document는 page\_content에, 원본 데이터의 모든 컬럼은 metadata에 저장한다.
2. 동재 재구성: 추론 시점, 검색된 문서의 metadata를 활용하여 LLM에게 전달할 최종 컨텍스트를 실시간으로 재구성한다. 이 과정에서 검색용 문서에는 없었던 신청 URL 등의 모든 정보를 포함시켜 자연스러운 문장으로 변환 후 전달한다.

이러한 전략을 통해, 검색의 정확도는 높이면서 LLM에게는 손실없는 원본 정보를 제공하여 답변의 충실성은 극대화할 수 있다. 이 전략은 LLM에게 전달되는 절대적인 컨텍스트 양이 많아지므로, Lost in the Middle 현상에 더욱 취약해지는 한계가 존재한다..

5.3. Lost in the Middle 현상 완화 전략

Lost in the Middle 현상은 LLM의 내재적인 한계이며, 동시에 해결이 까다로운 문제이기도 하다. 컨텍스트(Retrieved chunk)의 크기에 국한된 문제가 아니라 시스템 프롬프트와 chat history 등을 포함한 전체 입력 컨텍스트의 길이와 구조를 고려해야 하기 때문이다. 본 프로젝트에서는 이러한 Lost in the Middle 문제를 완화하기 위해 2가지 방식을 사용하였다.

LLM이 입력의 처음과 끝에 가장 높은 attention을 할당하는 것을 활용, 가장 중요한 정보인 컨텍스트를 chat history의 다음, 즉 프롬프트의 가능한 마지막 단에 배치하였다. 또한 chat history의 길이를 최근 2개의 대화로 제한하여 전체 프롬프트 길이를 최대한 짧게 유지하고, 핵심 정보가 중간에 묻힐 위험을 최소화 하였다.

그러나 이러한 방식은 상대적인 위치 조정으로 근본적인 해결이 되지 못한다. 경우에 따라선, 가장 유사도가 높은 첫 번째 문서가 되려 가장 중앙에 위치할 가능성이 존재한다. 따라서 좀 더 직접적이고 확실한 방안이 필요하다. 그 방안 중 하나는 바로 Re-ranker 모델의 도입이다. Re-ranker를 통해 문서들의 우선 순위를 산출하고 중요한 문서를 마지막단에 배치함으로써, LLM의 attention을 핵심 정보로 강제하는 방식이다. Re-ranker 모델의 도입은 본 프로젝트에서 다루지 못하였으나 향후 이를 적용하여 시스템을 고도화 할 계획이다..

Ⅵ. 결론 및 향후 연구

6.1. 결론

본 연구는 정형 데이터를 기반으로 고성능 RAG 시스템을 구축하기 위한 구체적인 아키텍처 최적화 전략을 제안하고 그 설계 사상을 기술하였다. 데이터의 본질적인 특성과 LLM의 내재적 한계를 분석하여, 검색용 데이터와 생성용 컨텍스트를 이원화하는 전략을 통해 검색의 정확성과 답변의 충실도라는 두 가지 목표를 동시에 추구하였다.

'의미론적 문서화'와 '청킹 배제' 전략으로 검색 정확도를 높이고, '동적 컨텍스트 재구성'으로 정보 손실 없는 답변 생성을 유도하였다. 또한, 'Lost in the Middle' 현상에 대응하기 위한 다층적 완화 전략을 수립함으로써, 단순히 RAG를 구현하는 것을 넘어 시스템의 내재적 한계를 극복하고자 하였다.

6.2. 연구의 한계점

본 연구에서 제안한 아키텍처의 우수성은 현재 벡터DB와 LLM의 일반적인 특성에 기반한 이론적 추론에 머물러 있다. 설계의 타당성을 객관적으로 입증하기에는 다음과 같은 명확한 한계가 존재한다.

첫째, 표준화된 평가 데이터셋의 부재이다. 제안한 아키텍처의 성능을 일관된 기준으로 측정하고 검증할 수 있는 질의응답(QA) 평가 세트가 없어, 구체적인 성과 확인이 어렵다.

둘째, 정량적·정성적 평가의 부재이다. 평가 데이터셋이 없으므로, 검색 성능 지표(e.g., MRR, Recall@K)나 생성된 답변의 품질에 대한 사용자 만족도 조사 등 객관적인 성능 평가를 수행하지 못하였다.

셋째, 원본 데이터의 내재적 비일관성이다. 본 연구는 정형 데이터의 틀을 사용했으나, 각 컬럼에 입력된 실제 텍스트의 형식이나 표현 방식이 정책마다 달라 데이터 정제에 상당한 노력이 소요되었다. 이는 전처리 로직의 복잡성을 가중시키는 요인으로 작용했으며, 본 연구에서는 이 문제를 수작업으로 해결하였으나 확장 가능한 시스템을 위해서는 자동화된 정제 파이프라인이 필요하다.

6.3. 향후 연구

앞서 언급한 한계점을 극복하고 제안 아키텍처의 실효성을 입증하기 위해, 다음과 같은 후속 연구를 진행하고자 한다.

1. 정량적 성능 평가를 위한 비교 실험 설계: 제안 방식의 우수성을 입증하기 위해, 다음과 같은 대조군을 설정하여 검색 및 답변 생성 성능을 비교 평가할 것이다.

* 대조군 1 (Raw Data Embedding): 원본 정형 데이터의 각 컬럼 값을 그대로 이어 붙여 임베딩하는 방식.
* 대조군 2 (Simple Key-Value Embedding): "정책명: OOO, 지원내용: XXX..." 와 같이 컬럼명을 포함한 Key-Value 형태로 변환하여 임베딩하는 방식.
* 실험군 (제안 방식): 본 연구에서 제안한 '의미론적 문서화' 방식.

2. Re-ranker 모델 도입을 통한 검색 정밀도 극대화: 'Lost in the Middle' 현상을 보다 근본적으로 해결하기 위해 Re-ranker 모델을 도입할 것이다. 이를 통해 가장 중요한 단일 문서를 선별하고 프롬프트의 최후미에 배치함으로써, LLM의 Attention을 핵심 정보로 강제하는 방식을 구현하여 시스템을 고도화할 계획이다.

6.4 회고

아쉬웠던 것

바이브 코딩 한 것 -> 복기하며 공부하기

문서 체인 다른 것도 써보기

RAG 기법들(multi query 등) 사용해보지 못한 것

~~6.4. 회고: '바이브 코딩'에서 데이터의 본질까지~~

~~솔직히 고백하자면, 이 프로젝트의 시작은 명확한 설계도보다는 "이렇게 하면 되지 않을까?"라는 직관, 이른바 '바이브 코딩'에 가까웠다. RAG의 일반적인 공식을 따르기보다, 데이터와 문제를 직접 마주하며 즉각적으로 해결책을 찾아 나갔다.~~

~~더불어, 이 과정에서 'Garbage In, Garbage Out'이라는 가장 근본적인 교훈을 다시 한번 깨달았다. 처음에는 정형화된 API 데이터이므로 깨끗할 것이라 막연히 기대했다. 하지만 현실은 달랐다. 같은 의미의 정보라도 정책마다 다른 형식과 표현으로 작성되어 있었고, 이는 \*\*'보이지 않는 비정형성'\*\*이었다. 결국 정교한 아키텍처를 설계하는 것만큼이나, 일관성 없는 데이터를 정제하고 씨름하는 과정에 많은 시간을 쏟아야 했다. 결국 뛰어난 RAG 시스템은 똑똑한 아키텍처뿐만 아니라, 양질의 데이터를 향한 집요한 노력 위에 세워진다는 것을 배울 수 있었다.~~

~~이 프로젝트는 나에게 기술적 성취뿐만 아니라, 직관을 논리로 다듬고, 화려한 기술 이면의 데이터 본질을 마주하는 엔지니어링의 즐거움을 알려주었다. 이 글이 또 다른 '바이브'로 무언가를 만들고 있는 개발자들에게 자신의 길을 확신하고 나아갈 작은 용기가 되기를 바란다~~.

References

[1] J. S. Chae, E. J. Hwang, S. M. Park, and Y. G. Hong. "Study on the use of open source-based simulator for SDV autonomous driving" KICS Magazine 41, no. 12, pp. 59-68